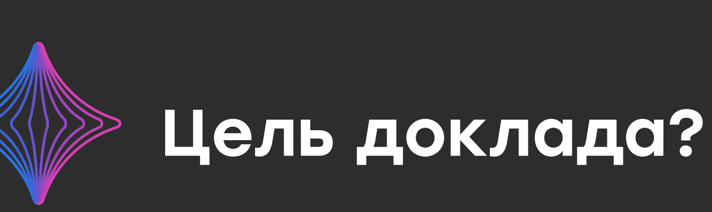
GraphQL Federation или как не выстрелить себе в ногу

Иван Решетин
Игорь Малюк

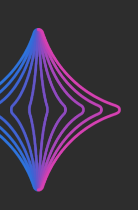
Юла.Платформа





Цель доклада?

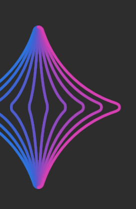




Цель доклада?

- Поделиться своим опытом

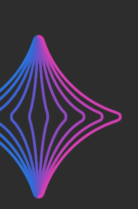




Цель доклада?

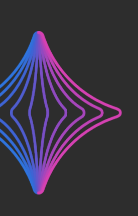
- Поделиться своим опытом
- Рассказать про наши ошибки





Содержание доклада

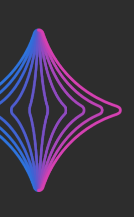




Содержание доклада

Что будет:



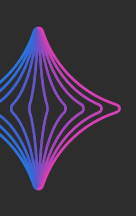


Содержание доклада

Что будет:

- Какие проблемы были до GraphQL



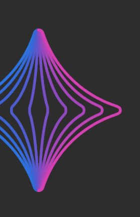


Содержание доклада

Что будет:

- Какие проблемы были до GraphQL
- Почему мы выбрали GraphQL



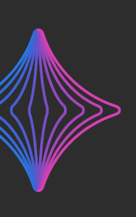


Содержание доклада

Что будет:

- Какие проблемы были до GraphQL
- Почему мы выбрали GraphQL
- Как мы переходили на федерацию



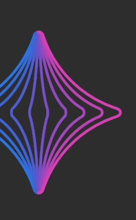


Содержание доклада

Что будет:

- Какие проблемы были до GraphQL
- Почему мы выбрали GraphQL
- Как мы переходили на федерацию
- С какими сложностями столкнулись





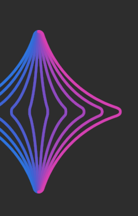
Содержание доклада

Что будет:

- Какие проблемы были до GraphQL
- Почему мы выбрали GraphQL
- Как мы переходили на федерацию
- С какими сложностями столкнулись

Чего не будет:





Содержание доклада

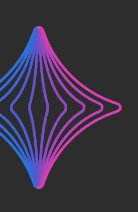
Что будет:

- Какие проблемы были до GraphQL
- Почему мы выбрали GraphQL
- Как мы переходили на федерацию
- С какими сложностями столкнулись

Чего не будет:

- Копи-паст документации





Содержание доклада

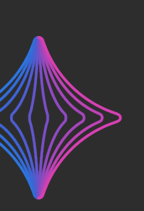
Что будет:

- Какие проблемы были до GraphQL
- Почему мы выбрали GraphQL
- Как мы переходили на федерацию
- С какими сложностями столкнулись

Чего не будет:

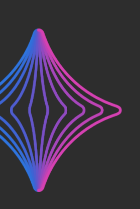
- Копи-паст документации
- Примеры кода





Что такое Юла?



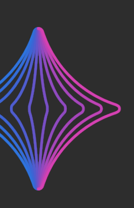


Что такое Юла?

Высокие нагрузки

27 млн	33 млн	200kRPS	120kRPS
MAU	объявлений	в Mongo	на бекенды



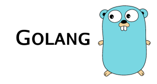


Что такое Юла?

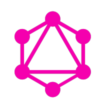
Высокие нагрузки

27 млн MAU	**33 млн** объявлений	**200kRPS** в Mongo	**120kRPS** на бекенды

Современные технологии











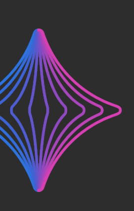
TARANTOOL









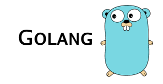


Что такое Юла?

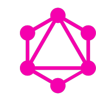
Высокие нагрузки

27 млн	33 млн	200kRPS	120kRPS
MAU	объявлений	в Mongo	на бекенды

Современные технологии











TARANTOOL







4 платформы

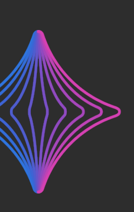










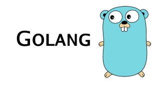


Что такое Юла?

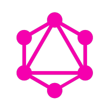
Высокие нагрузки

27 млн	33 млн	200kRPS	120kRPS
MAU	объявлений	в Mongo	на бекенды

Современные технологии










TARANTOOL





4 платформы





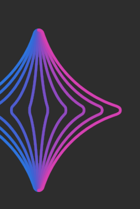


Масштабируемая архитектура

500 серверов **3 ДЦ**



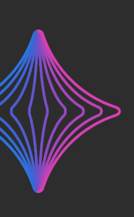

5
GraphQL
юла.tech



Что такое GraphQL?

GraphQL – это язык запросов для API



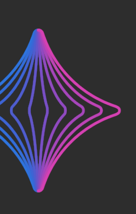


Что такое GraphQL?

GraphQL – это язык запросов для API

- Полное и понятное описание API



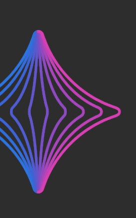


Что такое GraphQL?

GraphQL – это язык запросов для API

- Полное и понятное описание API
- Получать только необходимые данные





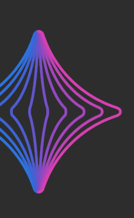
Что такое GraphQL?

GraphQL – это язык запросов для API

- Полное и понятное описание API
- Получать только необходимые данные

Типы операций





Что такое GraphQL?

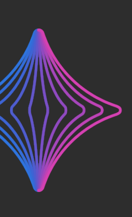
GraphQL – это язык запросов для API

- Полное и понятное описание API
- Получать только необходимые данные

Типы операций

- query





Что такое GraphQL?

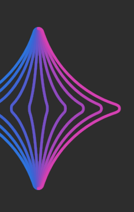
GraphQL – это язык запросов для API

- Полное и понятное описание API
- Получать только необходимые данные

Типы операций

- query
- mutation





Что такое GraphQL?

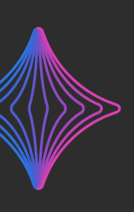
GraphQL – это язык запросов для API

- Полное и понятное описание API
- Получать только необходимые данные

Типы операций

- query
- mutation
- subscription





Что такое GraphQL?

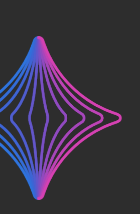
GraphQL – это язык запросов для API

- Полное и понятное описание API
- Получать только необходимые данные

Типы операций

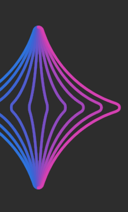
- query
- mutation
- subscription





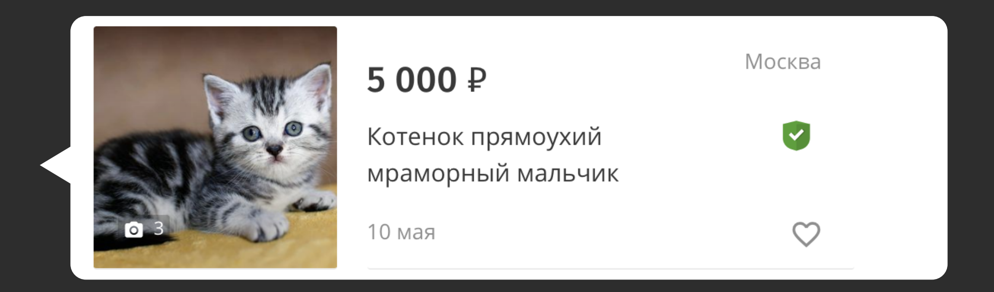
Что такое GraphQL схема



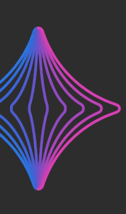


Что такое GraphQL схема

```
type Product {
    name: String
    images: [Image!]
    price: String
    city: String
    dateCreated: Timestamp
    dealBadge: boolean
}
```



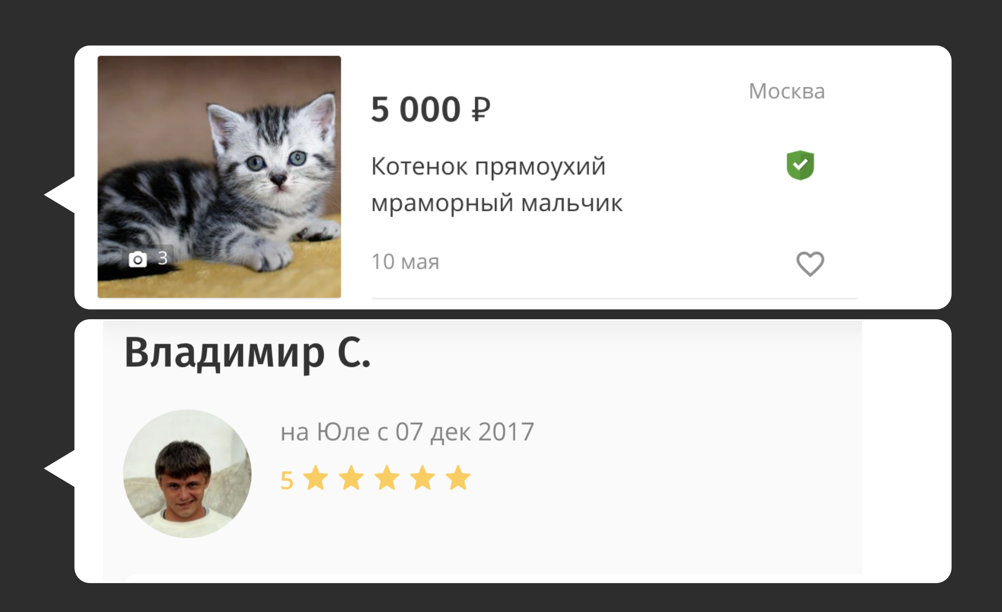




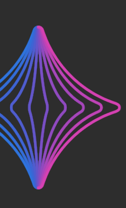
Что такое GraphQL схема

```
type Product {
   name: String
   images: [Image!]
   price: String
   city: String
   dateCreated: Timestamp
   dealBadge: boolean
}


type User {
   name: String
   image: Image!
   dateRegister: Timestamp
   rating: int
}
```

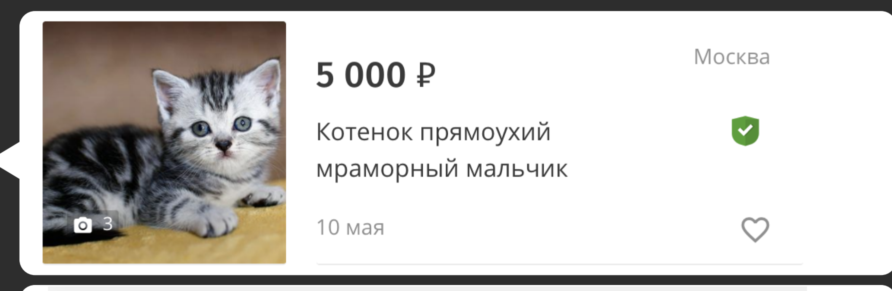




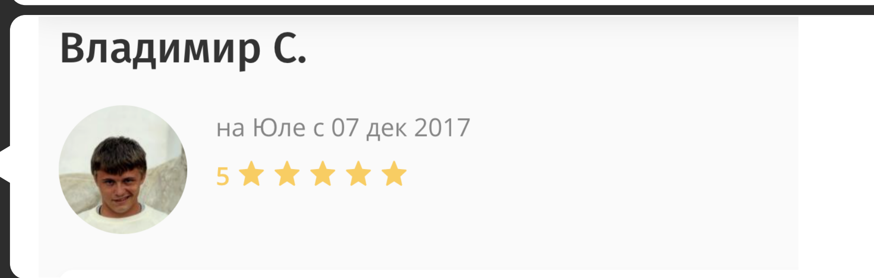


Что такое GraphQL схема

```
type Product {
    name: String
    images: [Image!]
    price: String
    city: String
    dateCreated: Timestamp
    dealBadge: boolean
}
```

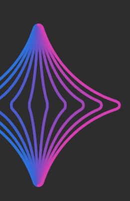



```
type User {
    name: String
    image: Image!
    dateRegister: Timestamp
    rating: int
}
```




```
type Image {
    url: String
}
```





GraphQL query

```
query {
  product {
      name
      price
      images {
        url
      }
      city
      user {
          name
          dateRegister
          rating
      }
  }
}
```

Котенок вислоухий мальчик

Добавить в избранное

6 000 ₽

Маша М. (6 объявл...

на Юле с 14 окт 2016

5.0 109 отз...

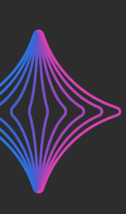
Подтвержденный профиль

VK проверено, 486 друзей

Купить

Безопасная сделка





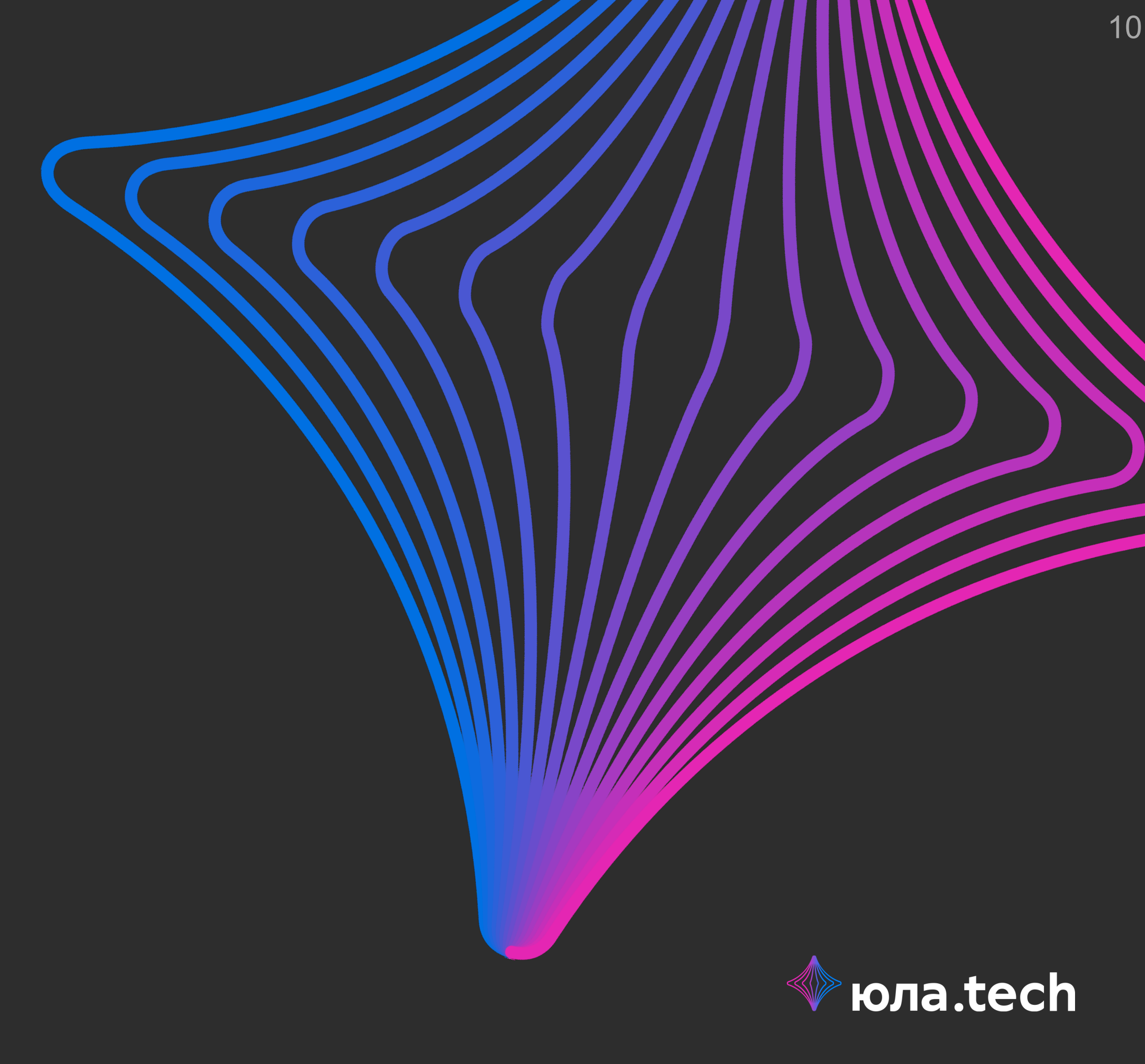
Исторически Юла - это REST

Опять хэйтят rest...



1

Увеличение отдаваемых данных



api/v1/products

```
Product :{
      id
      name
      description
      price
      images
      status
      subway
      boost_button
      info
      distance_text
}
```



api/v1/products

```
Product :{
      id
      name
      description
      price
      images
      status
      distance
      subway
      boost_button
      info
      price_text
      group_text
      distance_text
}
```



api/v1/products

```
Product :{
      id
      name
      description
      price
      category
      lat
      lon
      images
      status
      distance
      delivery
      complex
      subway
      boost_button
      info
      price_text
      group_text
      distance_text
}
```



api/v1/products

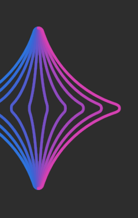
```
Product :{
        id
        name
        description
        price
        category
        lat
        lon
        images
        status
        distance
        delivery
        complex
        subway
        boost_button
        info
        price_text
        group_text
        distance_text
}
```



api/v2/products

```
Product :{
      id
      name
      description
      price
      category
      lat
      lon
      images
      status
      delivery
      boost_button
      price_text
      group_text
      distance_text
}
```





api/v2/products

```
Product :{
      id
      name
      description
      category
      lat
      lon
      images
      delivery
      complex
      boost_button
      info
      price_text
      group_text
      distance_text
      can_buy
      delivery_status
      date_created
}
```



api/v2/products

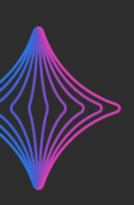
```
Product :{
      id
      name
      description
      category
      lat
      lon
      images
      delivery
      complex
      boost_button
      info
      price_text
      group_text
      distance_text
      can_buy
      delivery_status
      date_created
      is_active
      order
      user
}
```



api/v2/products

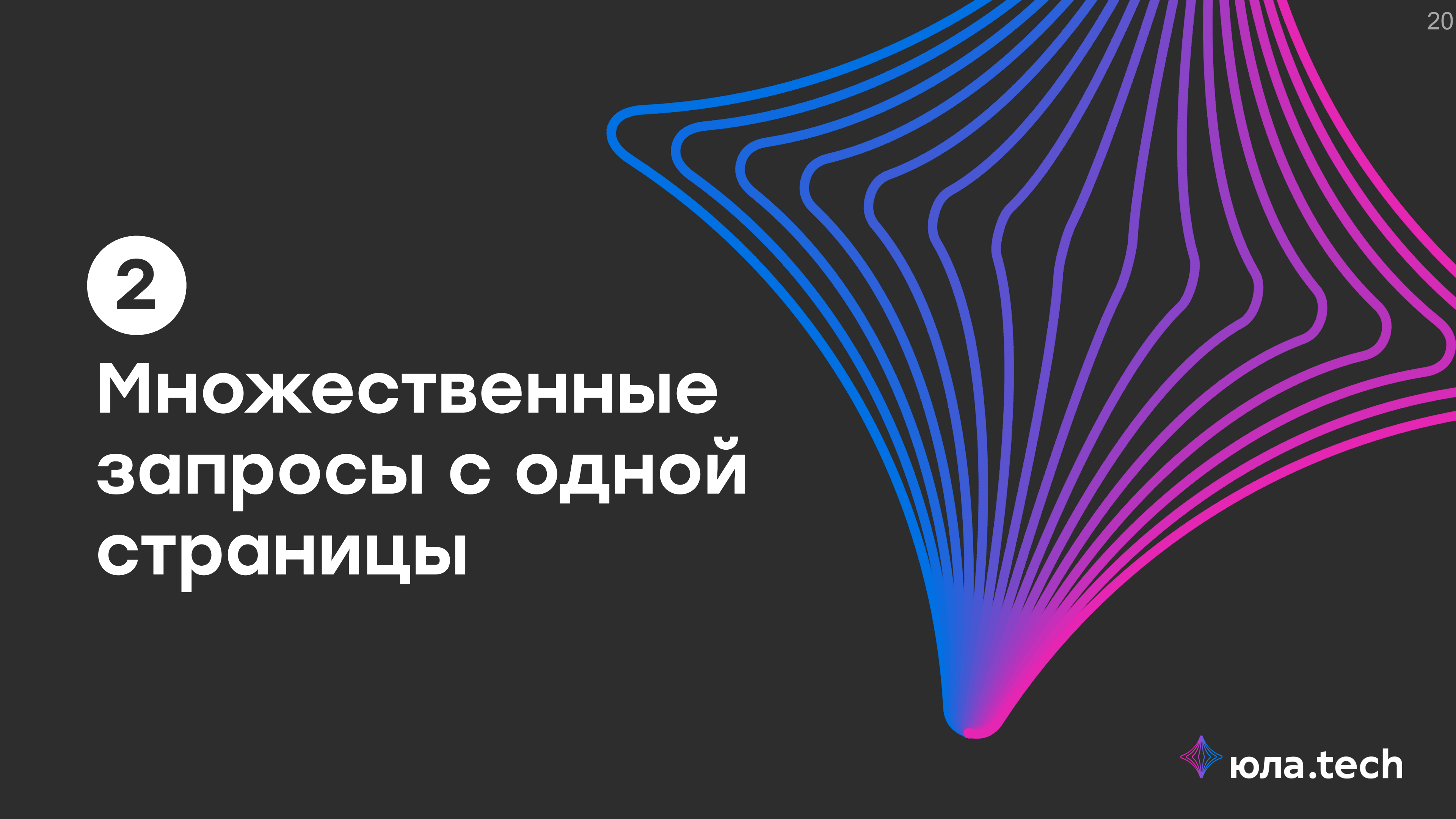
```
Product :{
      id
      name
      description
      category
      lat
      lon
      images
      delivery
      complex
      boost_button
      info
      price_text
      group_text
      distance_text
      can_buy
      delivery_status
      date_created
      is_active
      order
      user
}
```

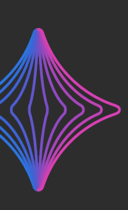




api/v2/products




2
Множественные запросы с одной страницы
юла.tech



Главная страница

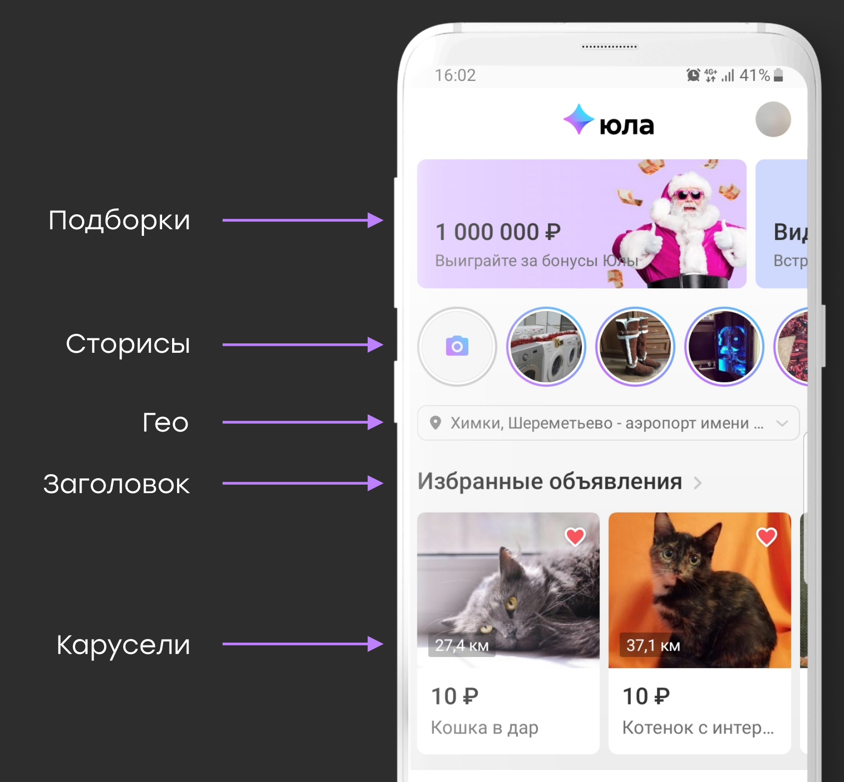
Подборки

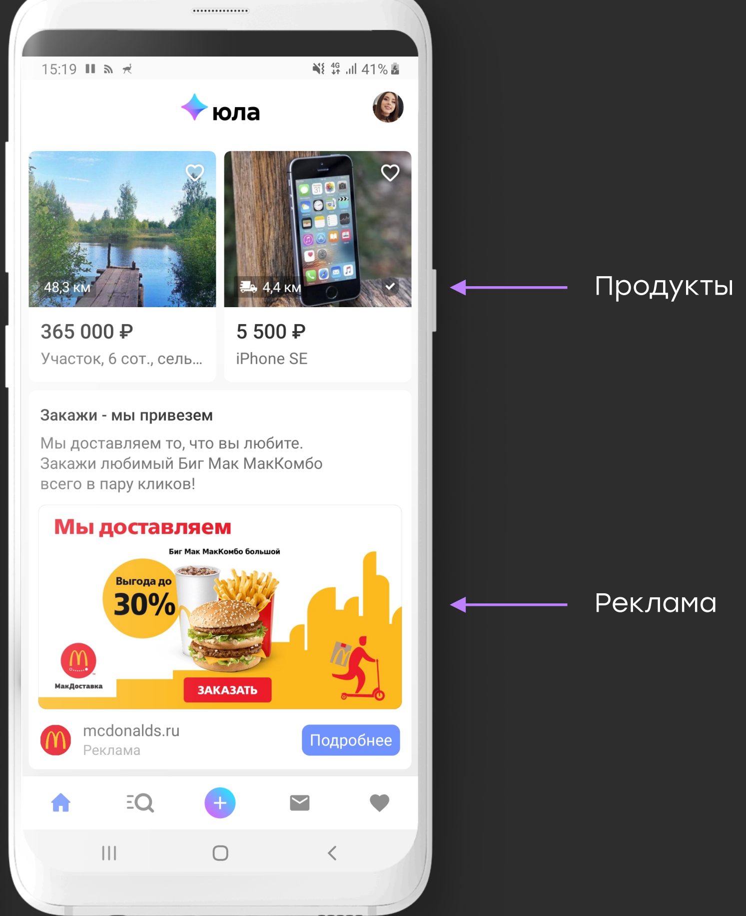
Сторисы

Гео

Заголовок

Карусели

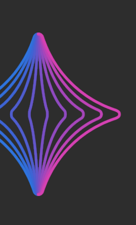




Продукты

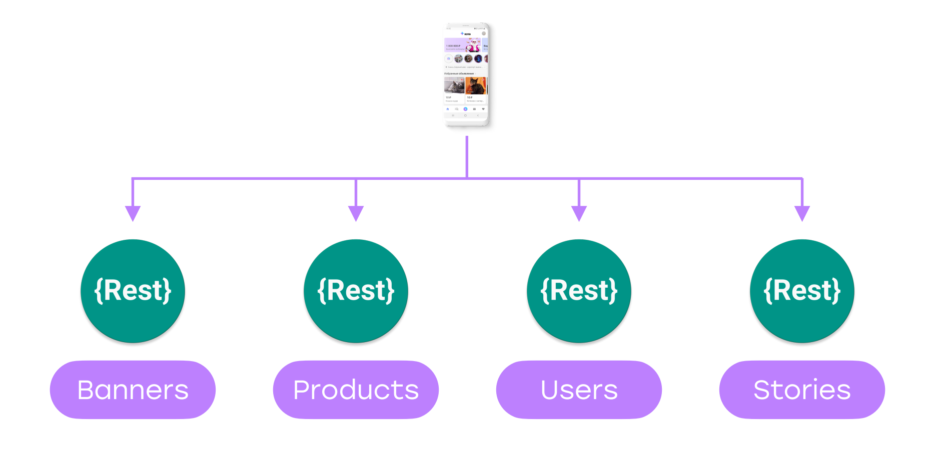
Реклама





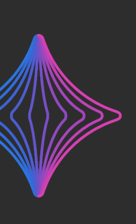
Варианты запроса главной страницы

- Несколько REST запросов



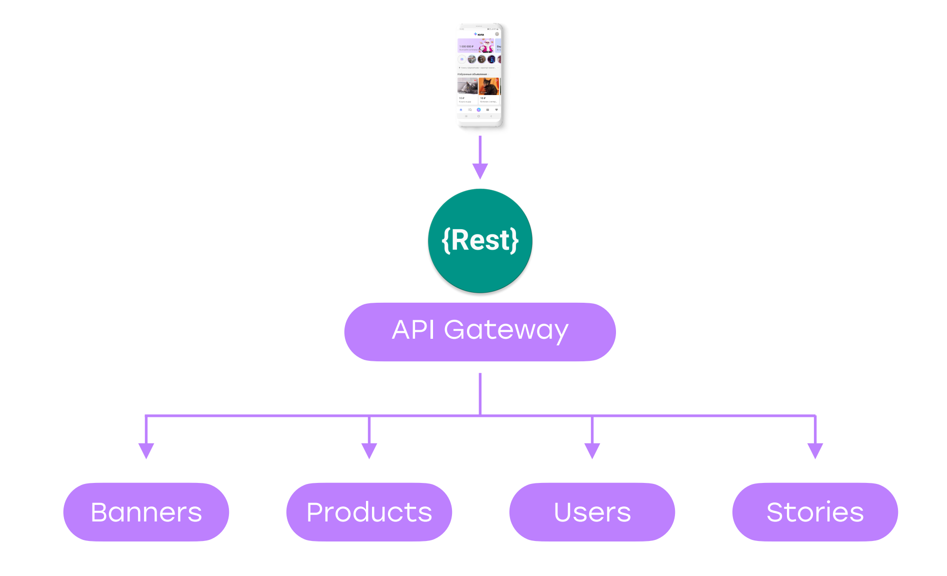
Минусы:

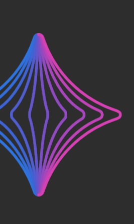
- Все проблемы REST
- Много запросов с клиентов
- Походы напрямую в сервис



Варианты запроса главной страницы

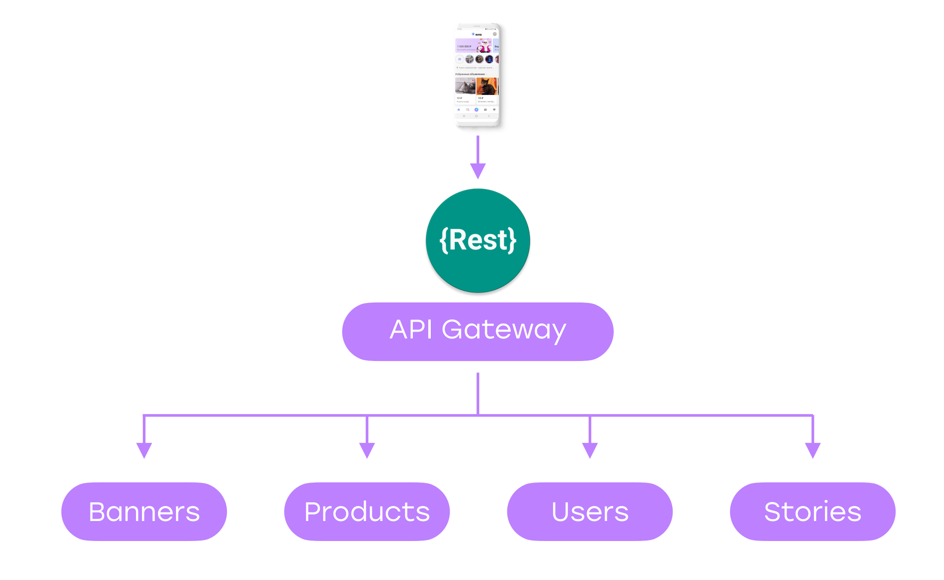
- Несколько REST запросов
- API Gateway





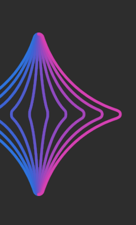
Варианты запроса главной страницы

- Несколько REST запросов
- API Gateway



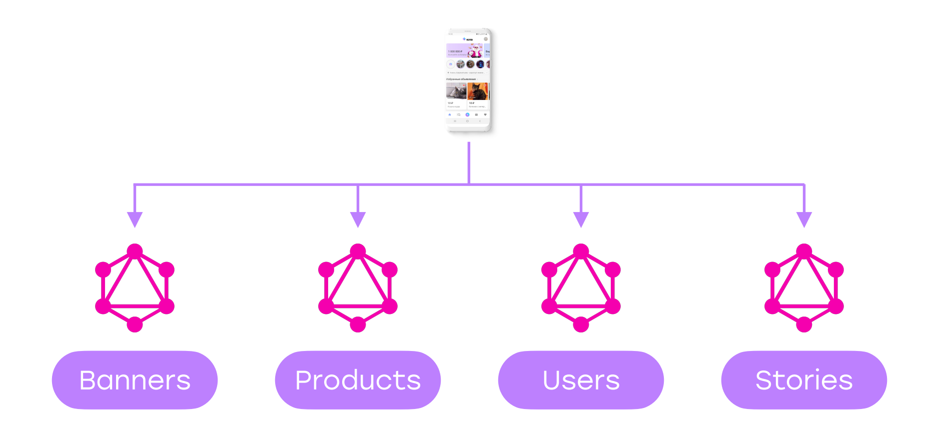
Минусы:

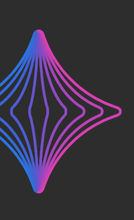
- Все проблемы REST
- Дополнительный слой



Варианты запроса главной страницы

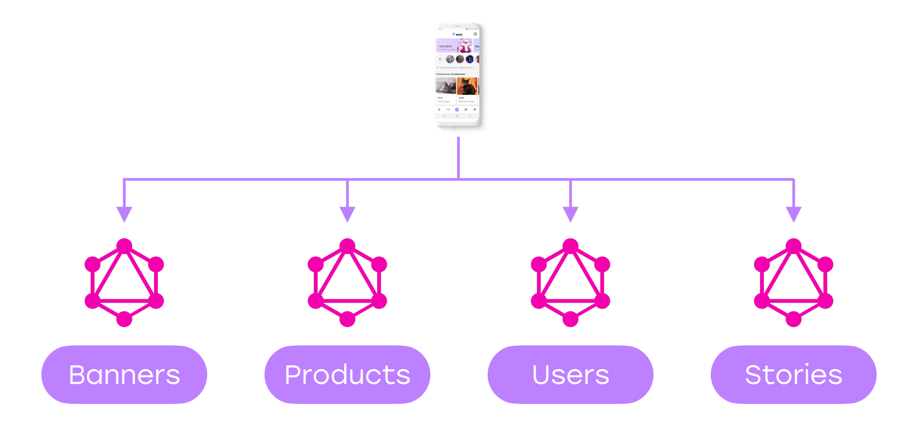
- Несколько REST запросов
- API Gateway
- Несколько GraphQL





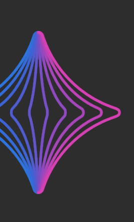
Варианты запроса главной страницы

- Несколько REST запросов
- API Gateway
- Несколько GraphQL



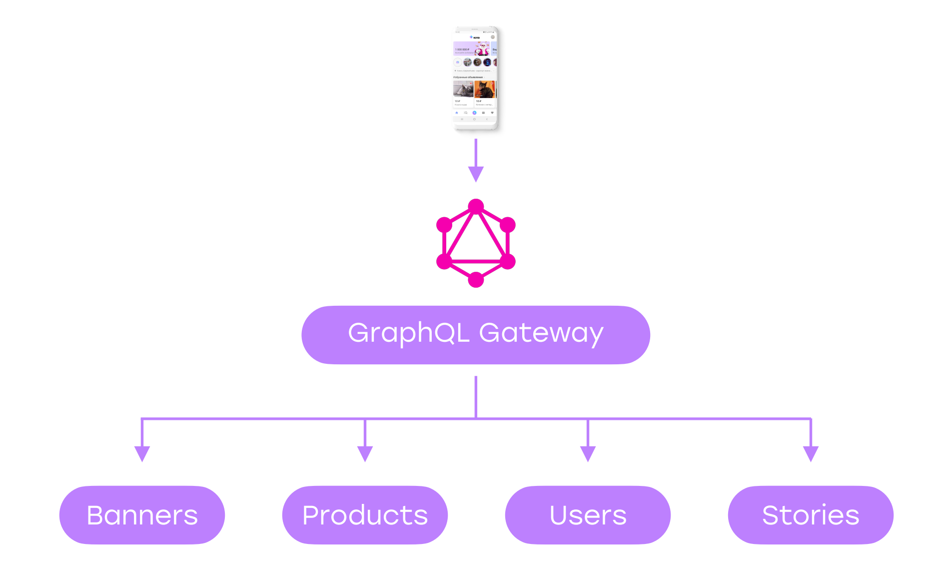
Минусы:

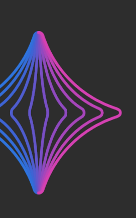
- Конфликты типов
- Усложняется поддержка
- Не подходит для большого проекта



Варианты запроса главной страницы

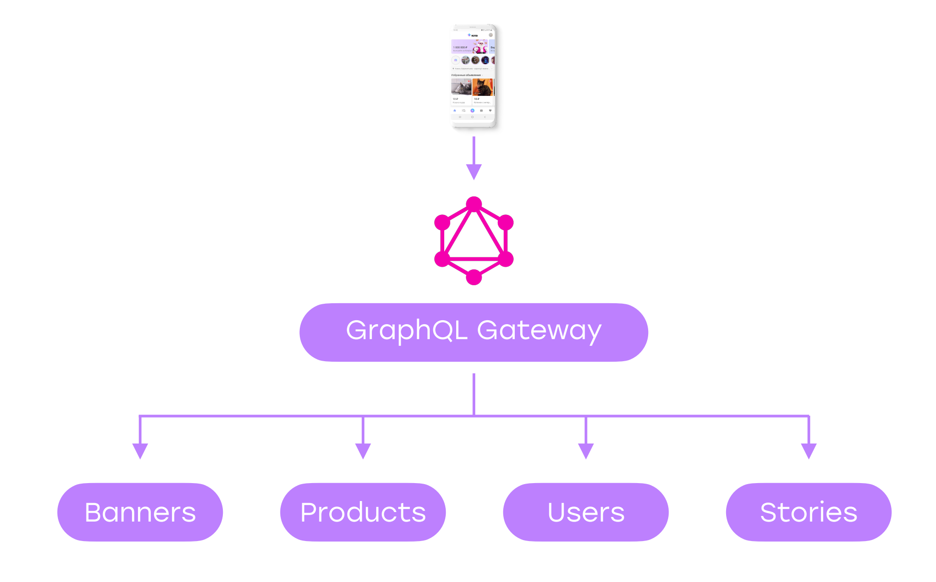
- Несколько REST запросов
- API Gateway
- Несколько GraphQL
- GraphQL Gateway





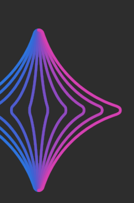
Варианты запроса главной страницы

- Несколько REST запросов
- API Gateway
- Несколько GraphQL
- GraphQL Gateway

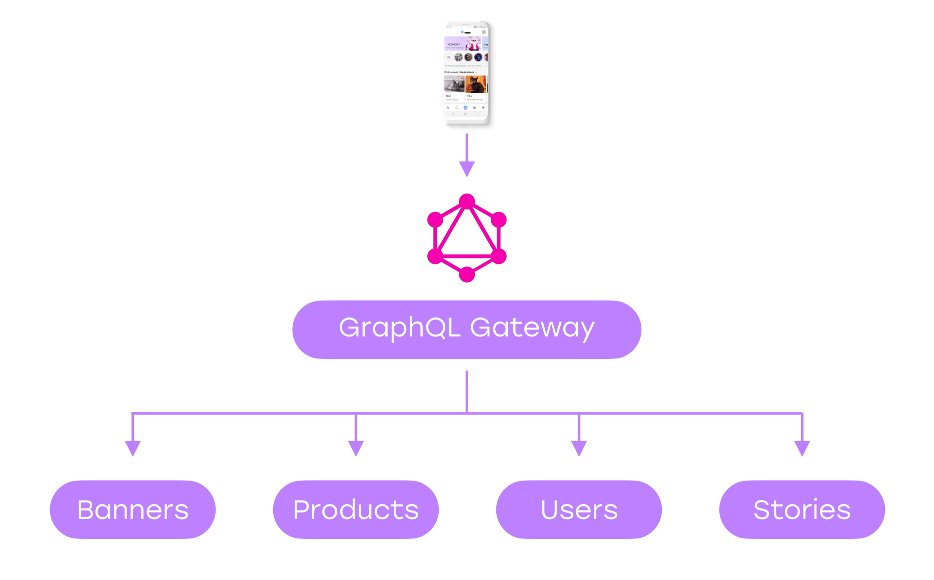


Минусы:

- Дополнительный слой
- Больше нет :)

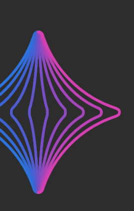


Выбираем GraphQL!



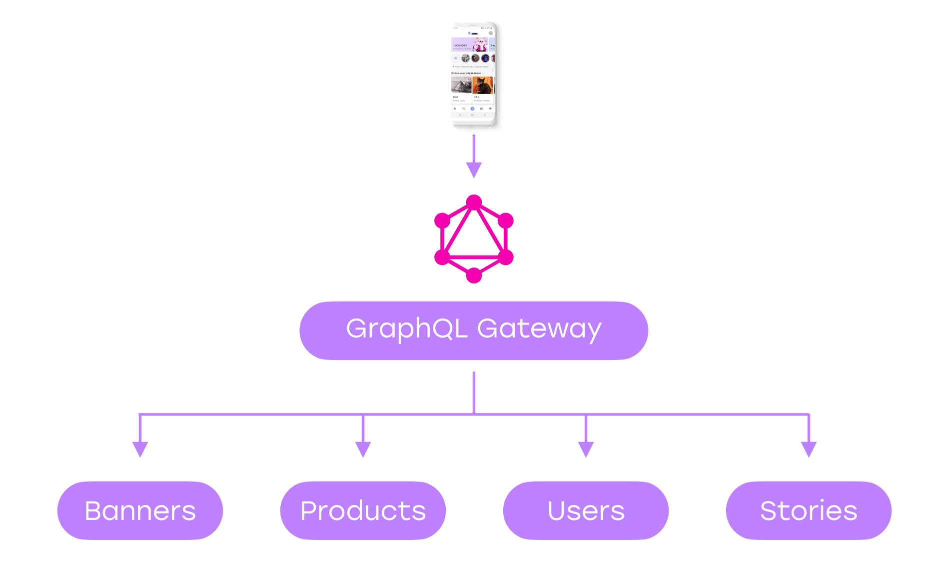
Минусы:

- Расширяем тех стек



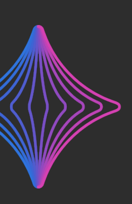
Выбираем GraphQL!

- Разные клиенты – разные отображения



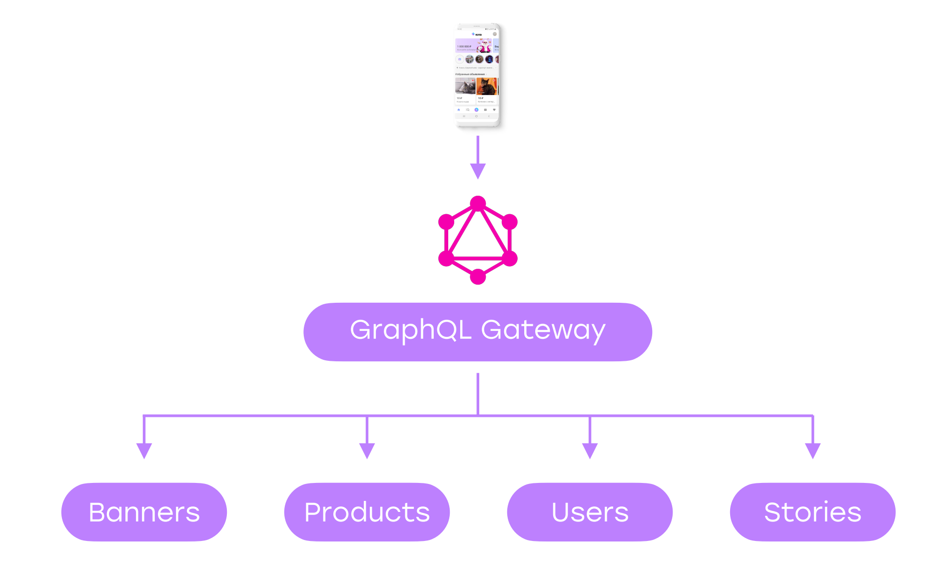
Минусы:

- Расширяем тех стек



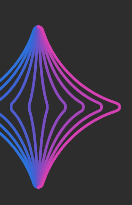
Выбираем GraphQL!

- Разные клиенты – разные отображения
- Статистика запрашиваемых полей



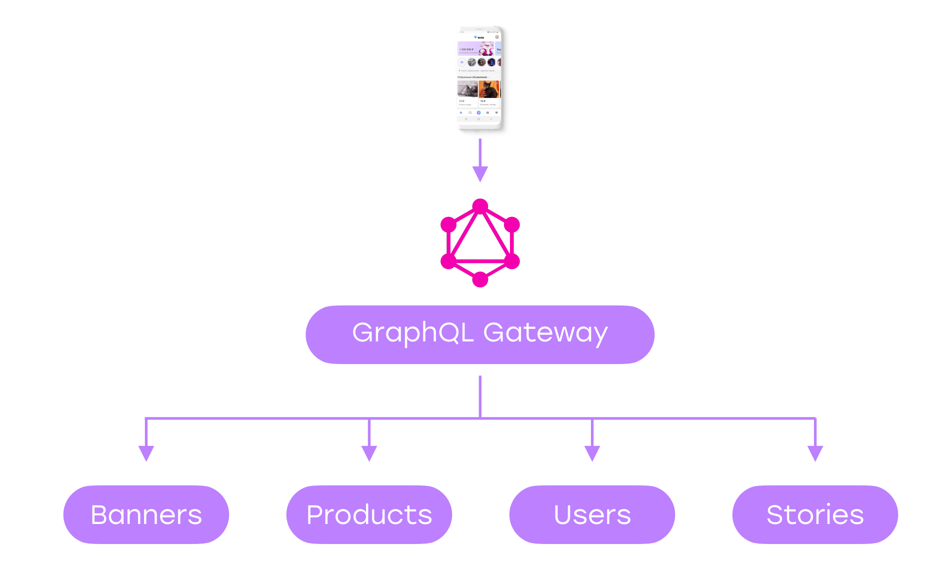
Минусы:

- Расширяем тех стек



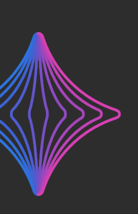
Выбираем GraphQL!

- Разные клиенты – разные отображения
- Статистика запрашиваемых полей
- Скрытие реализации целевых сервисов

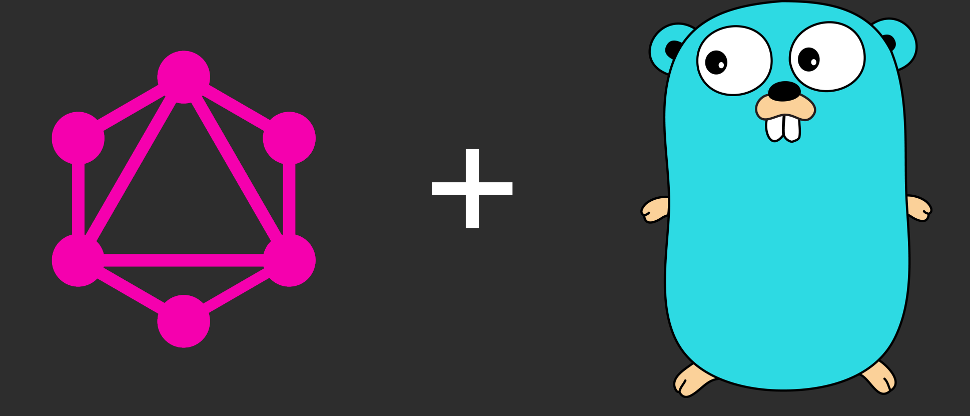


Минусы:

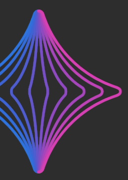
- Расширяем тех стек



-Какую библиотеку выбрать?
-Ту которая «Schema First»!







Schema First

```graphql
// products.graphql
extend type Query {
    products(limit: Int): [Product!]
    product(id: ID!): Product
}

# Базовый продукт/объявление
type Product {
    id: ID!
    url: String!
    owner: User!
    name: String!
}
```

```go
// models.gen.go
package generated

type Product struct {
        ID    string `json:»id"`
        URL   string `json:"url"`
        Owner *User  `json:"owner"`
        Name  string `json:"name"`
}
```

```go
/products.go
.vers

 will be automatically regenerated based on the schema, any resolver
ons
opied through when generating and any unknown code will be moved to the end.

ontext"
nt"
roducts/generated"
roducts/models"

luctResolver) Owner(ctx context.Context, obj *models.Product) (*models.User,

nic(fmt.Errorf("not implemented"))

ryResolver) Products(ctx context.Context, limit *int) ([]*models.Product,

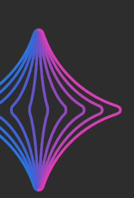
nic(fmt.Errorf("not implemented"))

ryResolver) Product(ctx context.Context, id string) (*models.Product, error)

nic(fmt.Errorf("not implemented"))
}

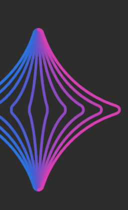
// Product returns generated.ProductResolver implementation.
func (r *Resolver) Product() generated.ProductResolver { return &productResolver{r} }

type productResolver struct{ *Resolver }
```



Плюсы Schema First

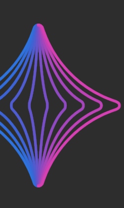
gqlgen



Плюсы Schema First

- Меньше ручного труда

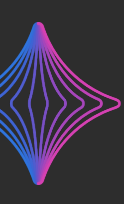
gqlgen



Плюсы Schema First

- Меньше ручного труда
- Описываем схему вместе с клиентами

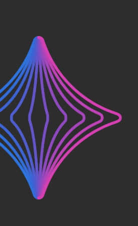
gqlgen



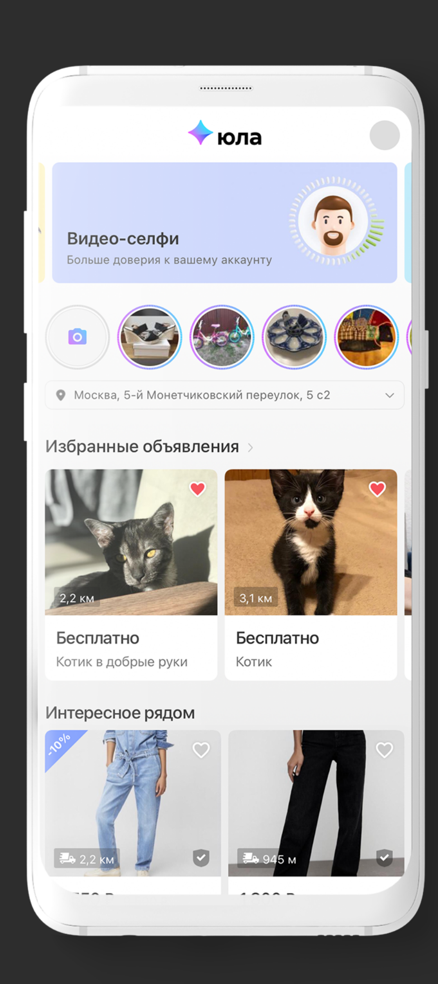
Плюсы Schema First

- Меньше ручного труда
- Описываем схему вместе с клиентами
- На выходе готовые mock'и

gqlgen



Главная – один запрос за всеми данными





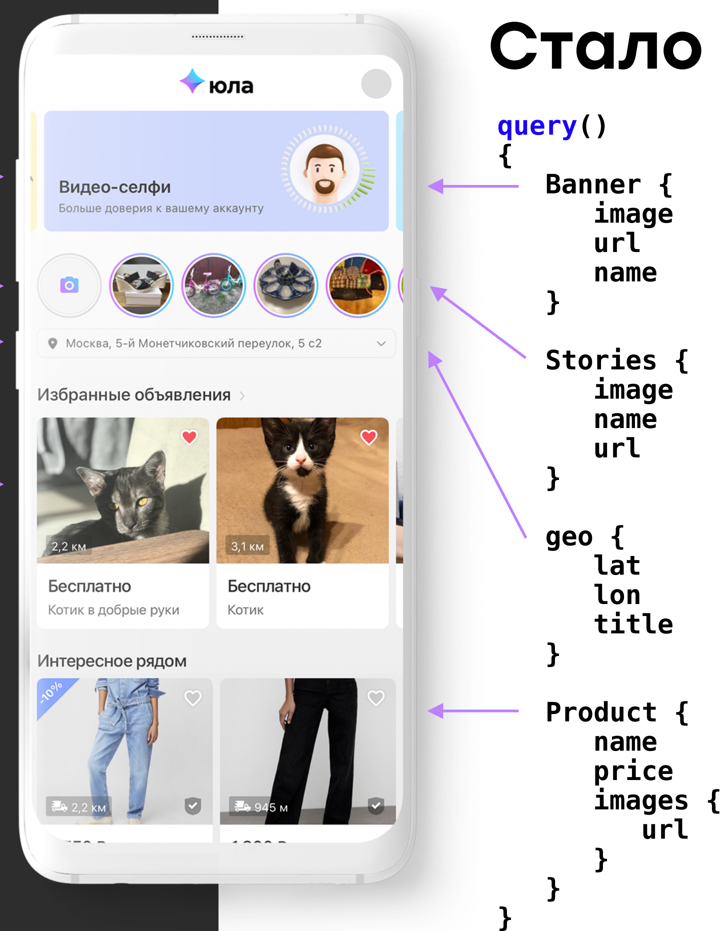
Было

[GET] /api/v1/banners

[GET] /api/v1/stories

[GET] /api/v1/geo

[GET] /api/v1/products



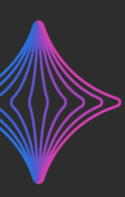
Стало

```
query()
{
   Banner {
      image
      url
      name
   }

   Stories {
      image
      name
      url
   }

   geo {
      lat
      lon
      title
   }

   Product {
      name
      price
      images {
         url
      }
   }
}
```



GraphQL схема при первом запуске

Query
init Init!
feed Feed!

Init
version ErrorVersion
configApp ConfigApp!

ErrorVersion
message String!
title String!
code Int!

ConfigApp
splits String!
abTests JsonObject
features Features!

Features
authVerify Boolean!

Feed
items [FeedItem!]
pageInfo PageInfo

FeedItem
<<union>>
possible types
CarouselItem
ProductItem
PromotedProductItem
AdvertItem
HeaderItem
BundleItem

PageInfo
threshold Int!
cursor Cursor!
hasNextPage Boolean!

CarouselItem
type String!
offsetItem Int!
carousel Carousel!

AdvertItem
type String!
offsetItem Int!

HeaderItem
type String!
offsetItem Int!
title String!

BundleItem
type String!
offsetItem Int!
bundle [Bundle!]!

Bundle
eventId Int!
id String!
name String!
description String!
images [String!]!
action JsonObject
pixelUrls [String!]

Carousel
position Int!
clientParams JsonObject
title String!
carouselId String!
qid String!
order Int!
settings CarouselSettings!
items [CarouselItems!]!
action JsonObject

CarouselSettings
can_delete Boolean!
has_feed Boolean!
feed_limit Int!
preview_limit Int!
type String!
alias String!

CarouselItems
<<union>>
possible types
ProductItem
UserItem

UserItem
user User

ProductItem
type String!
offsetItem Int!
product Product!
offsetProduct Int!
productAnalytics ProductAnalytics!

PromotedProductItem
type String!
offsetItem Int!
productPromoted Product!
productPromotedFields ProductPromotedFields
displayType Int!
offsetProduct Int!
bannerId Int
productAnalytics ProductAnalytics!

Product
id String!
owner User!
linkedId String
type String!
name String!
price ProductPrice!
datePublished Timestamp!
isVerified Boolean!
isPromoted Boolean!
images [Image!]!
location Location!
category Int!
subcategory Int!
favorite ProductFavorite!
distanceText String!
deliveryAvailable Boolean!
paymentAvailable Boolean!
badge Badge
description String
salaryText String
employerText String

ProductAnalytics
isSold Boolean!
soldMode Int!
isDeleted Boolean!
isBlocked Boolean!
isArchived Boolean!
archiveMode Int!
isExpiring Boolean!
blockMode Int!
engine String
promotionType Int!
showUrls [String]
clickUrls [String]

ProductPromotedFields
<<union>>
possible types
VacancyProductPromotedFields

VacancyProductPromotedFields
employerText String!
salaryText String!

User
id String!
name String!
isVerified Boolean!
image Image
displayPhoneNum String
ratingMark Float
settings UserSettings

Image
id String!
num Int!
url String!

UserSettings
displayPhone Boolean!

ProductPrice
realPrice Price!
realPriceText String!
origPrice Price!
origPriceText String!
discount Int
discountText String!

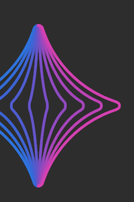
Price
price Int!

Location
latitude Float!
longitude Float!
description String!
city String!
city_name String!
addressText String

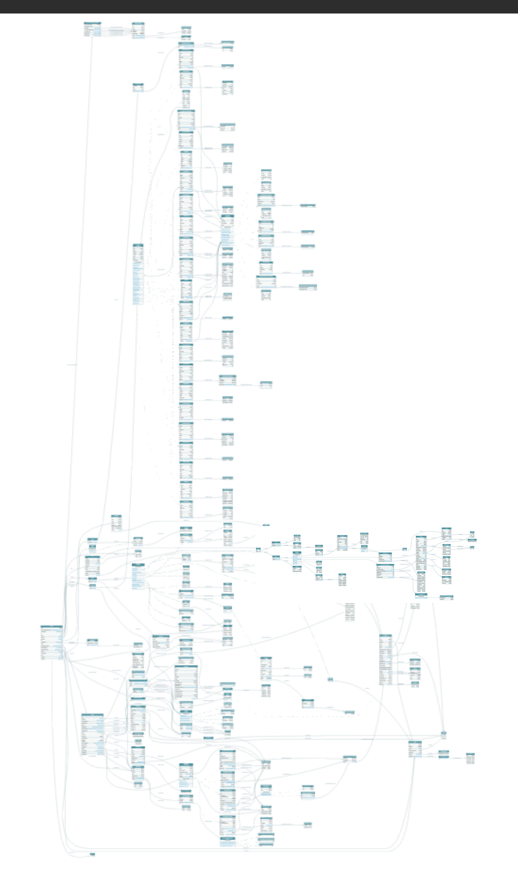
ProductFavorite
productId String!
enabled Boolean!
timestamp Timestamp!

Badge
title String
description String
color_background String
color_text String



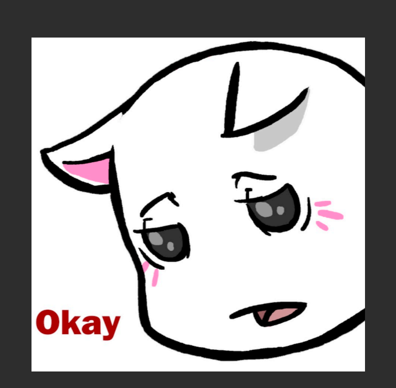


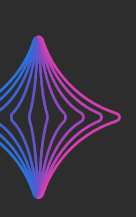
...1.5 года спустя



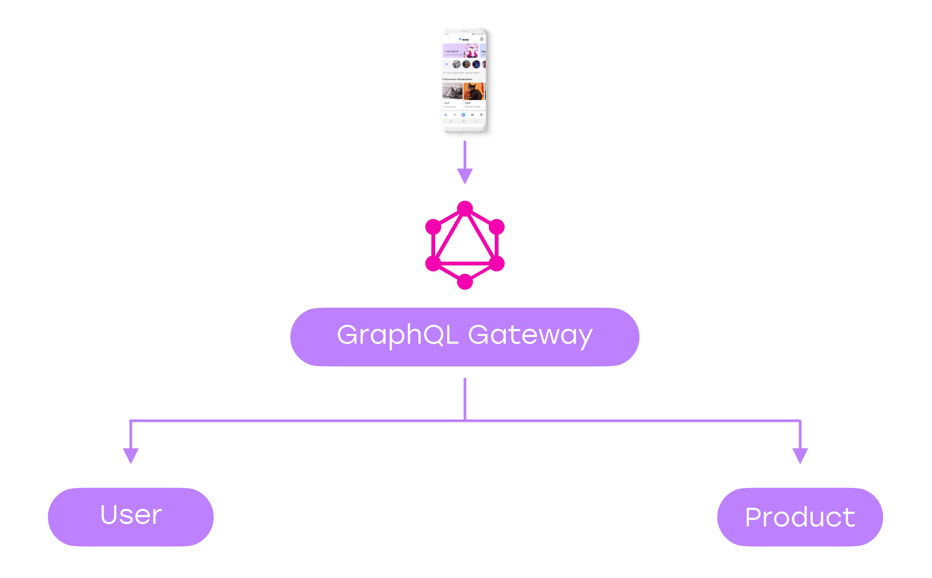


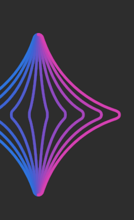
Проблемы?
...да





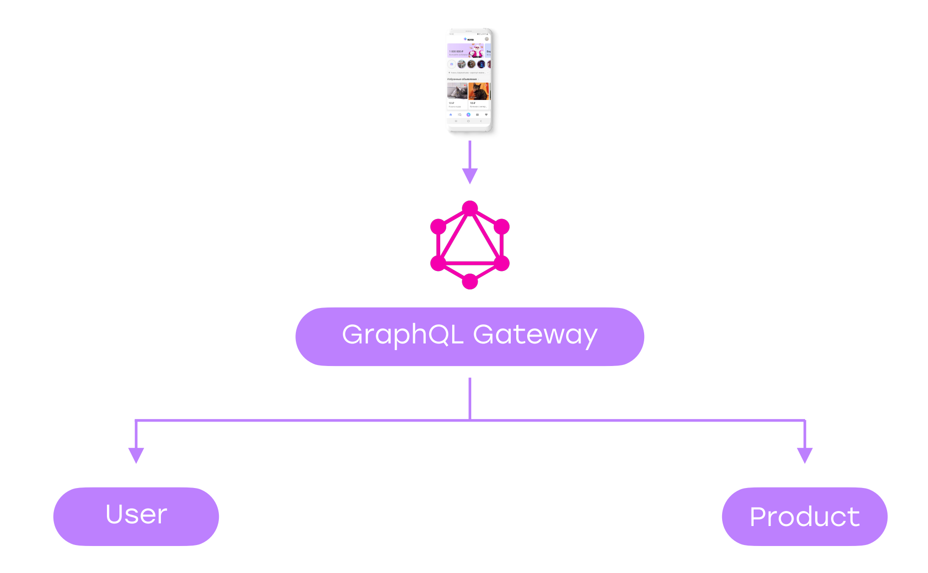
Бутылочное горлышко:

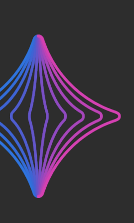




Бутылочное горлышко:

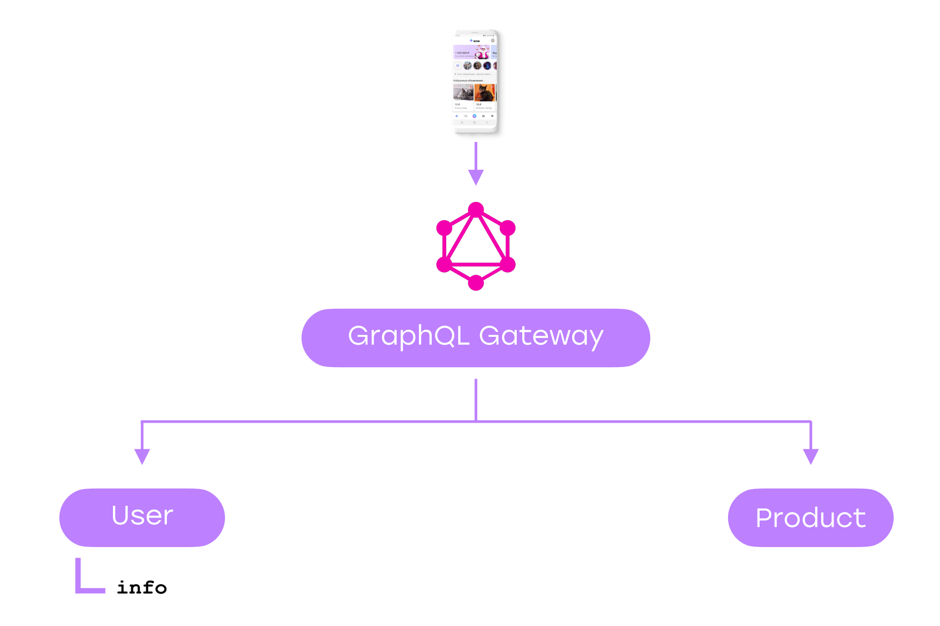
- Gateway и сервисы – разные команды

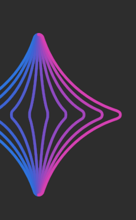




Бутылочное горлышко:

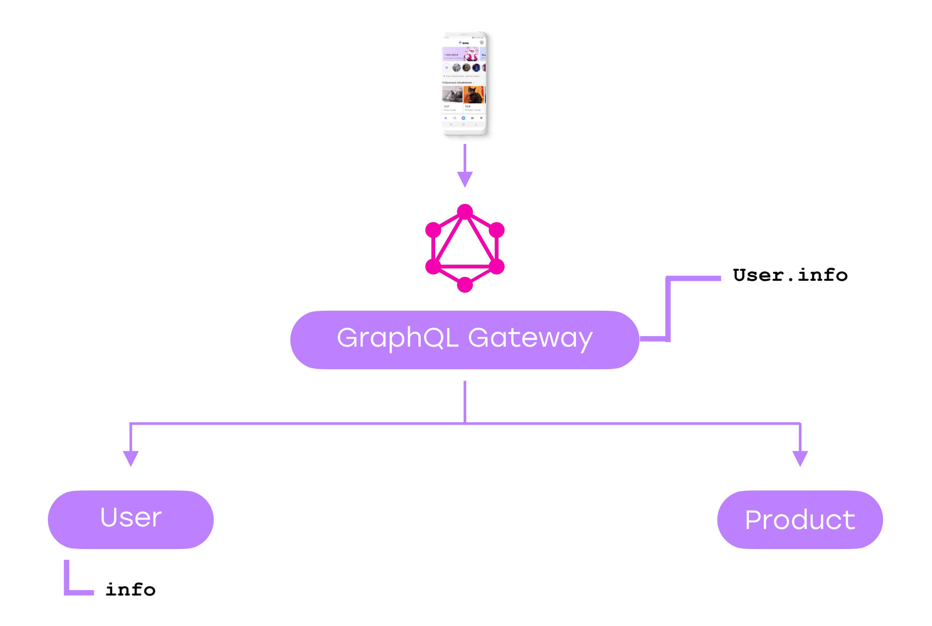
- Gateway и сервисы – разные команды

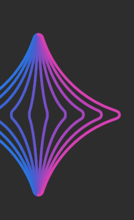




Бутылочное горлышко:

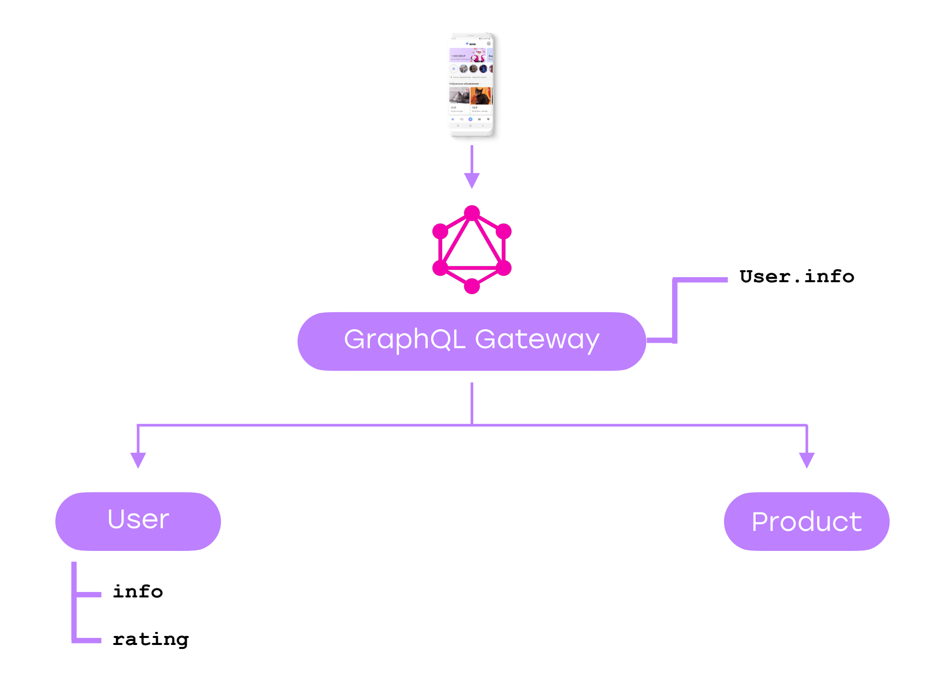
- Gateway и сервисы – разные команды

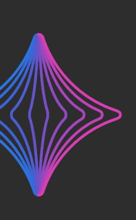




Бутылочное горлышко:

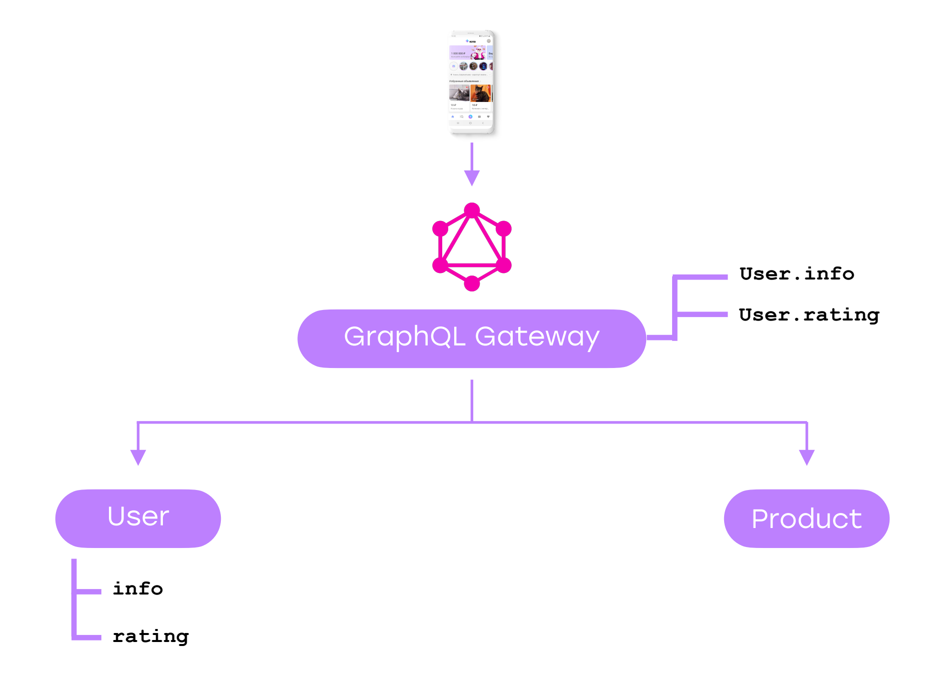
- Gateway и сервисы – разные команды

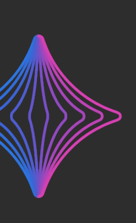




Бутылочное горлышко:

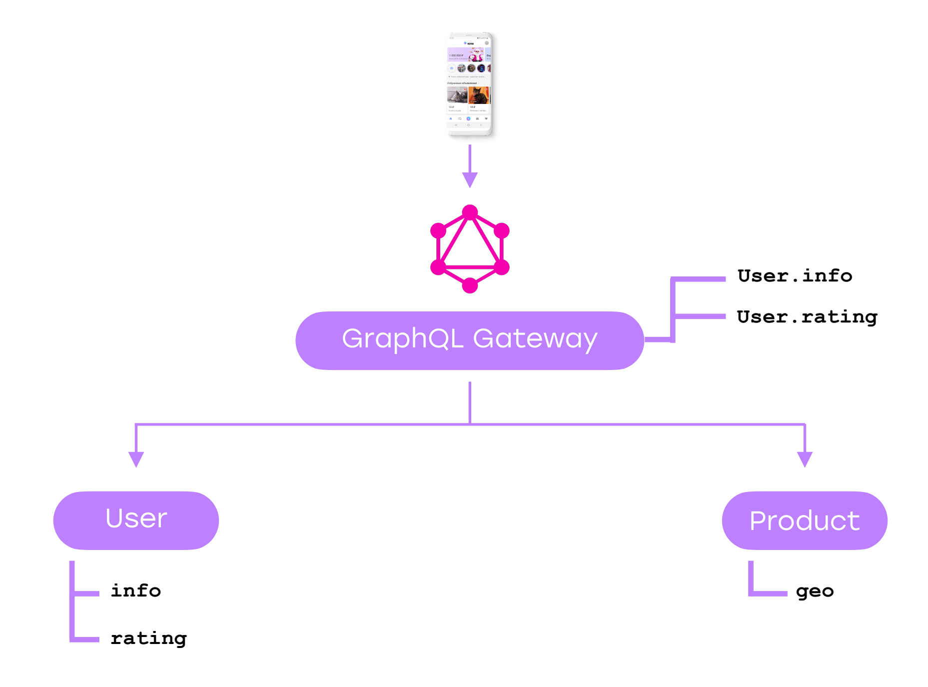
- Gateway и сервисы – разные команды

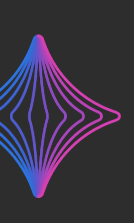




Бутылочное горлышко:

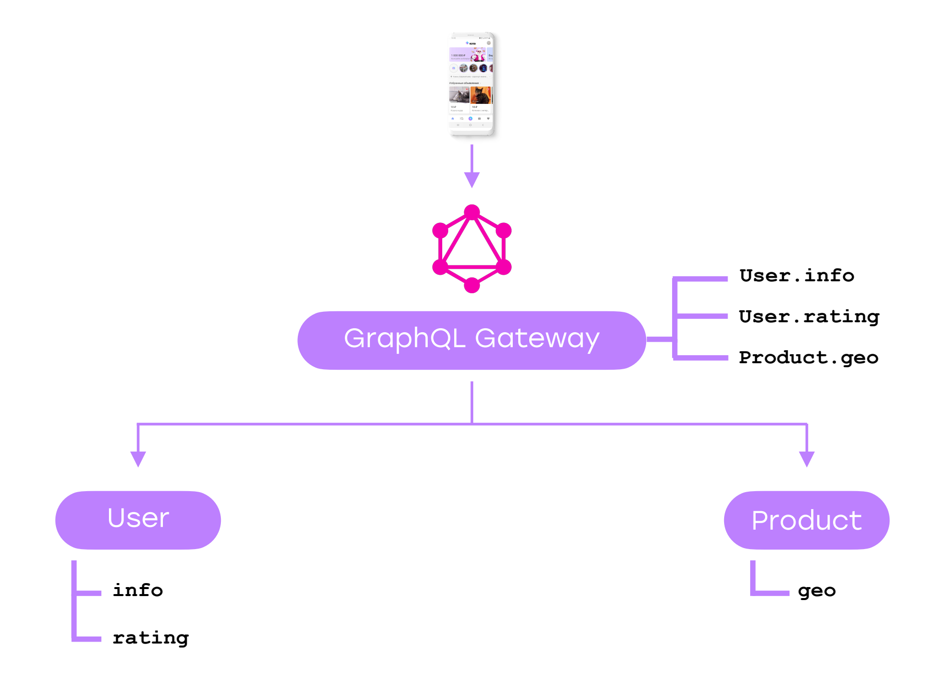
- Gateway и сервисы – разные команды

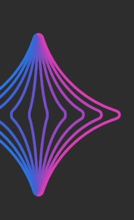




Бутылочное горлышко:

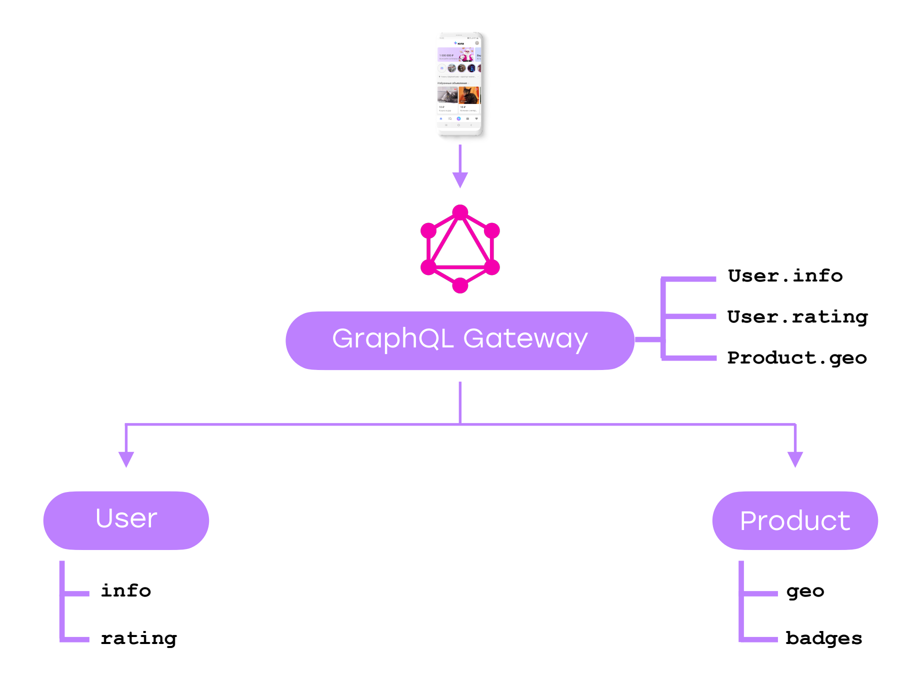
- Gateway и сервисы – разные команды

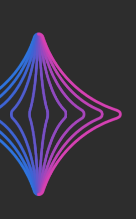




Бутылочное горлышко:

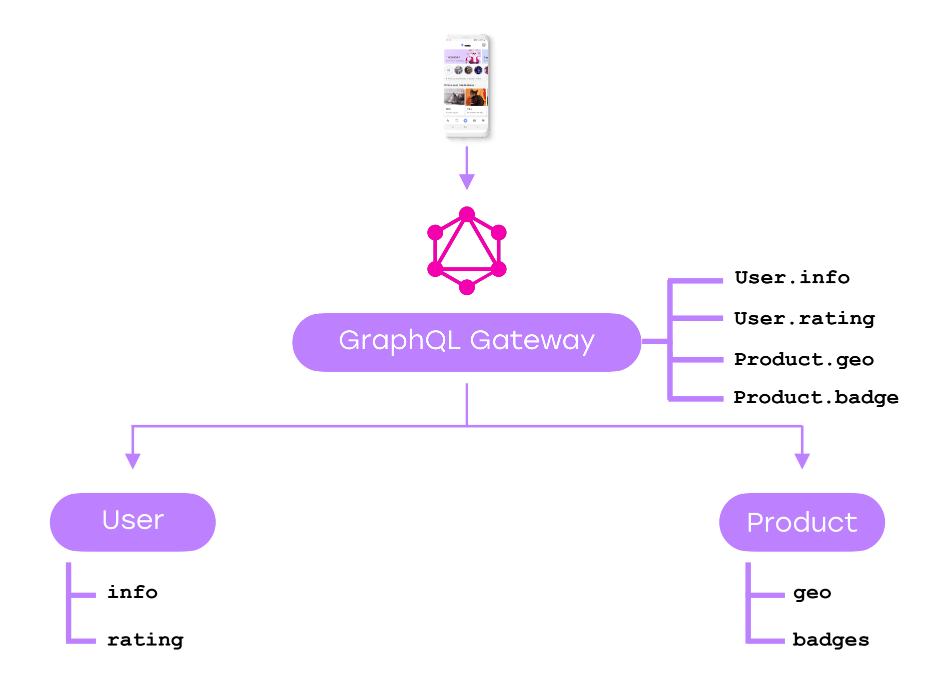
- Gateway и сервисы – разные команды

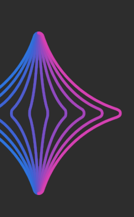




Бутылочное горлышко:

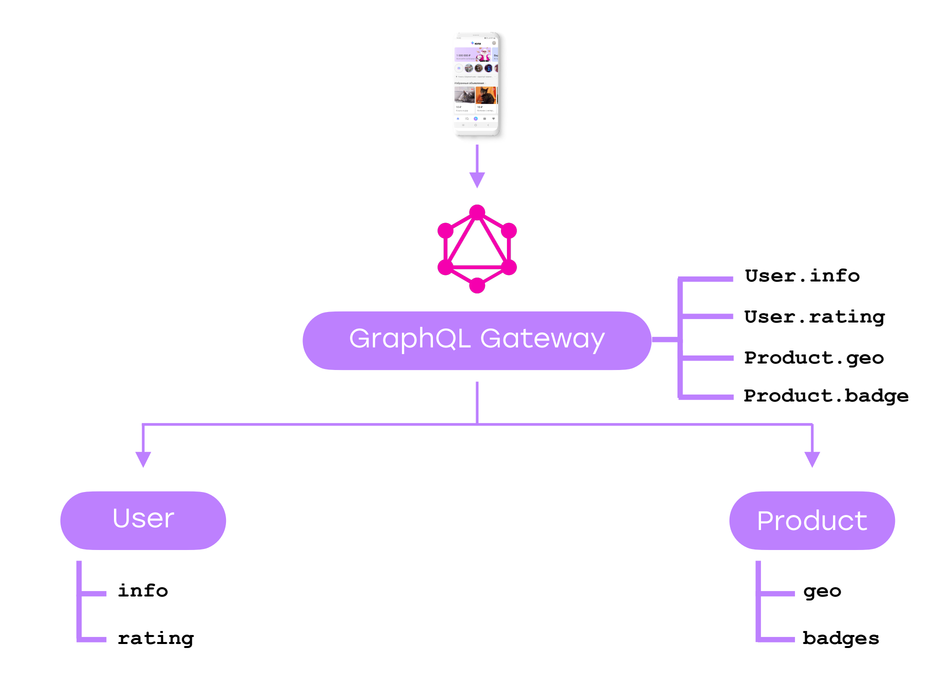
- Gateway и сервисы – разные команды

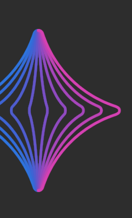




Бутылочное горлышко:

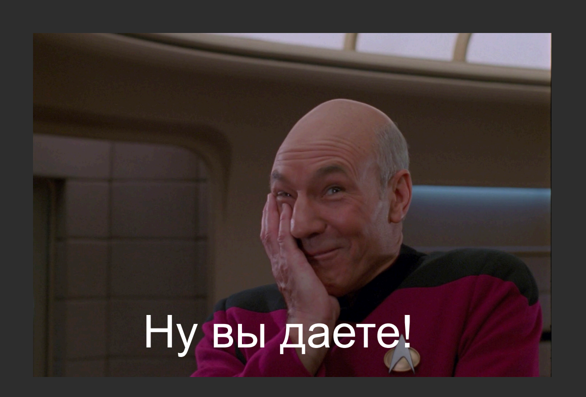
- Gateway и сервисы – разные команды
- Gateway содержит бизнеслогику

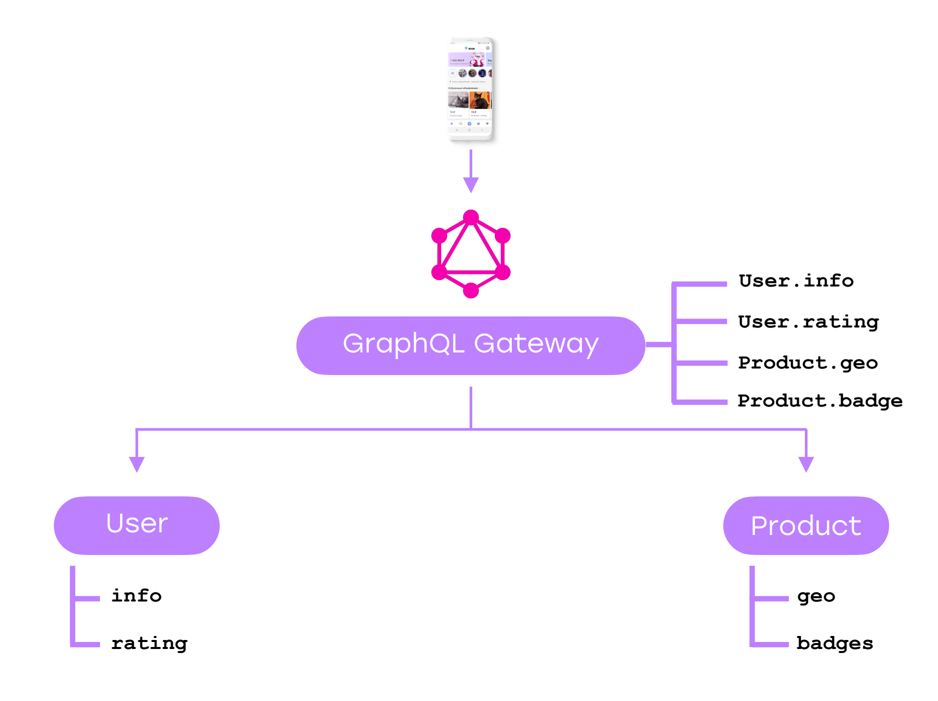


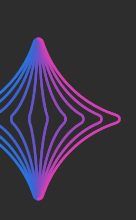


Бутылочное горлышко:

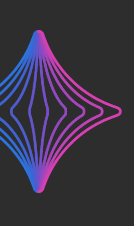
- Gateway и сервисы – разные команды
- Gateway содержит бизнеслогику







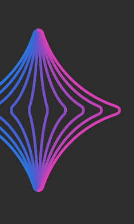
Как будем решать проблему?



Как будем решать проблему?

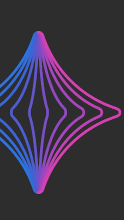
- Смириться и терпеть





Как будем решать проблему?

- Смириться и терпеть
- Открыть полный доступ



Как будем решать проблему?

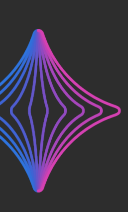
- Смириться и терпеть
- Открыть полный доступ
- Правки через ревью

Наташа, посмотри PR

Мы всё переделали

Ещё и сквошнули

Вообще всё, честно!

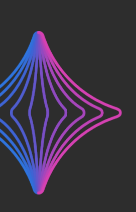


Как будем решать проблему?

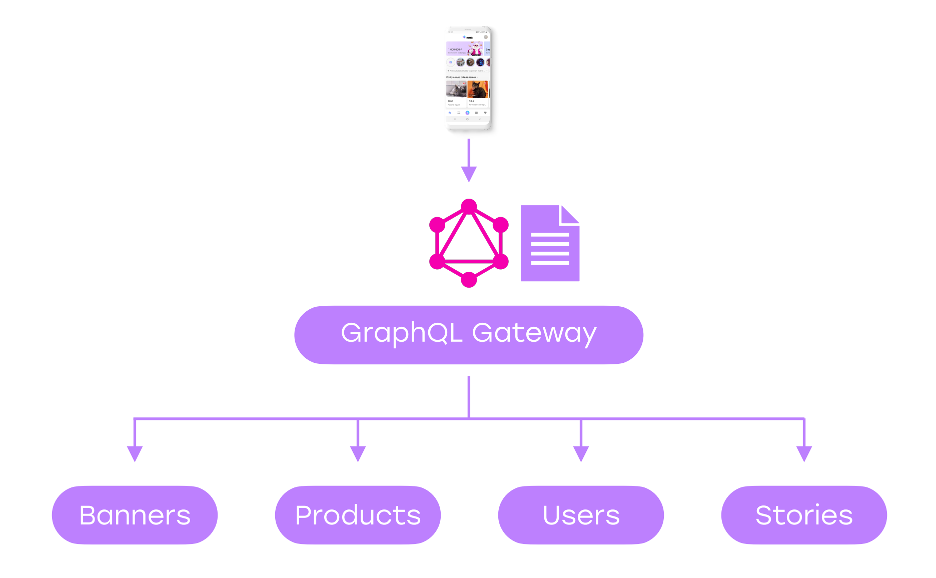
- Смириться и терпеть
- Открыть полный доступ
- Правки через ревью
- Найти серебряную пулю (для нашей ноги)

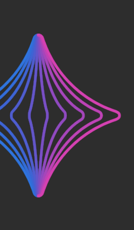
Нам ~~нужен план~~ нужны новые требования!

юла.tech



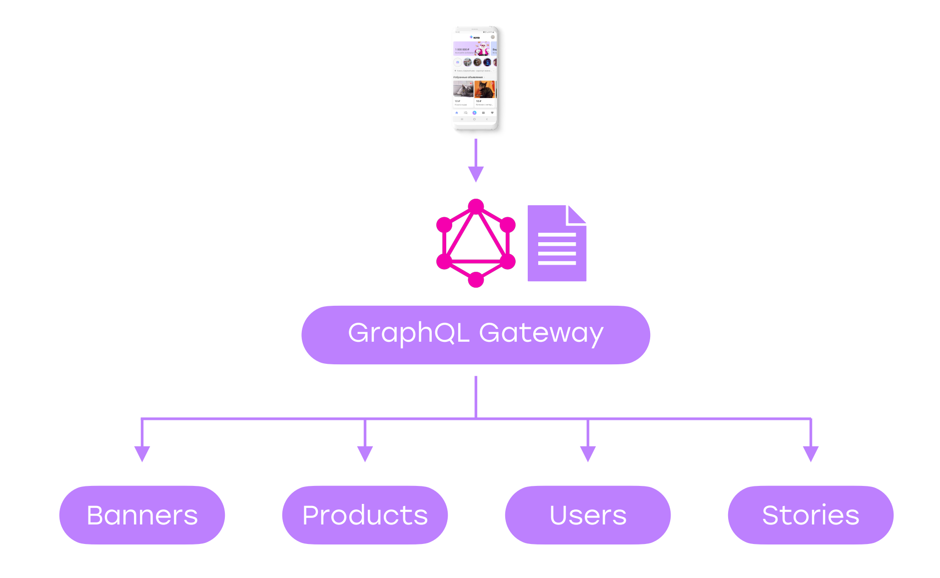
Что мы хотели получить?

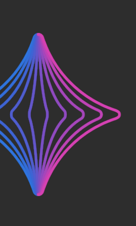




Что мы хотели получить?

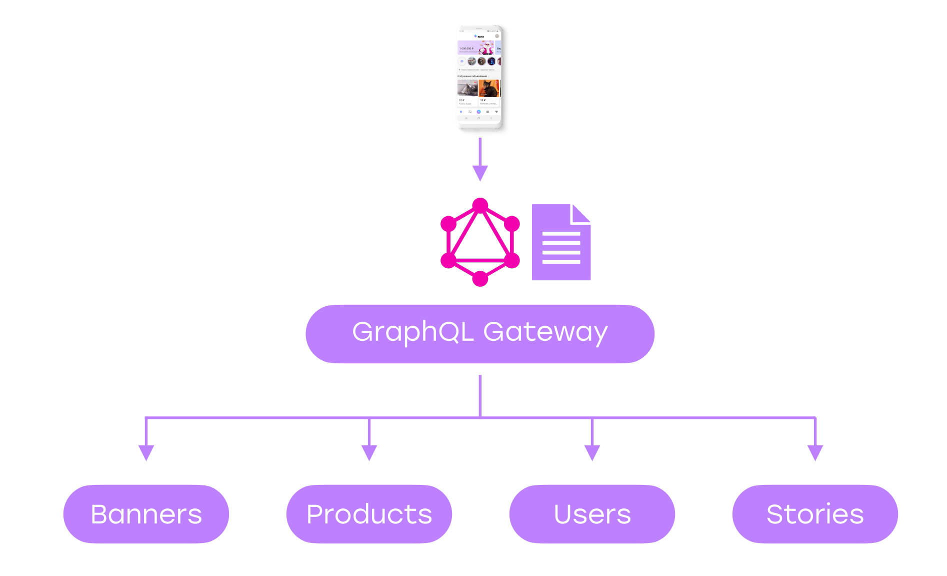
- Одна точка входа для клиентов

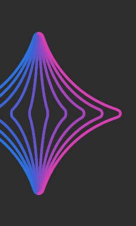




Что мы хотели получить?

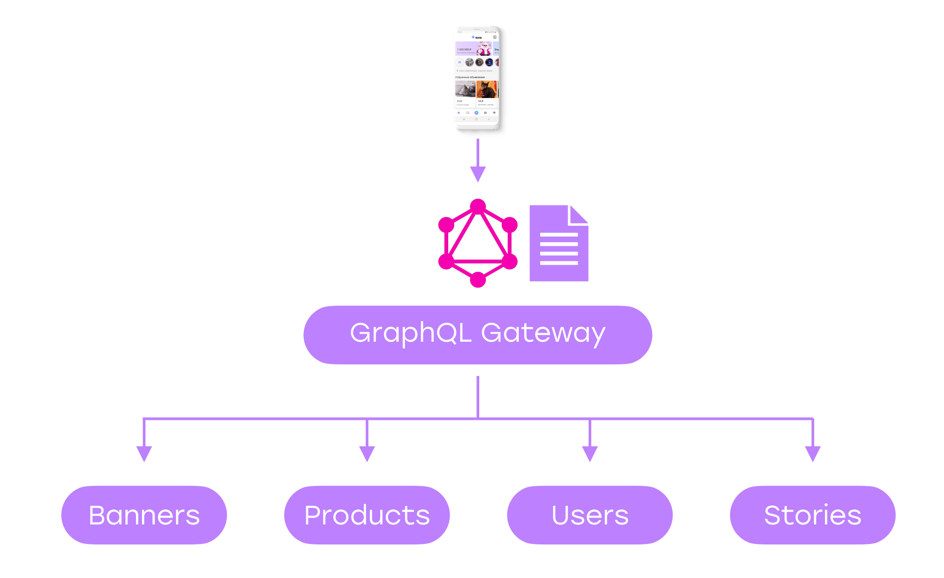
- Одна точка входа для клиентов
- Единая схема для клиентов

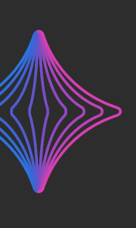




Что мы хотели получить?

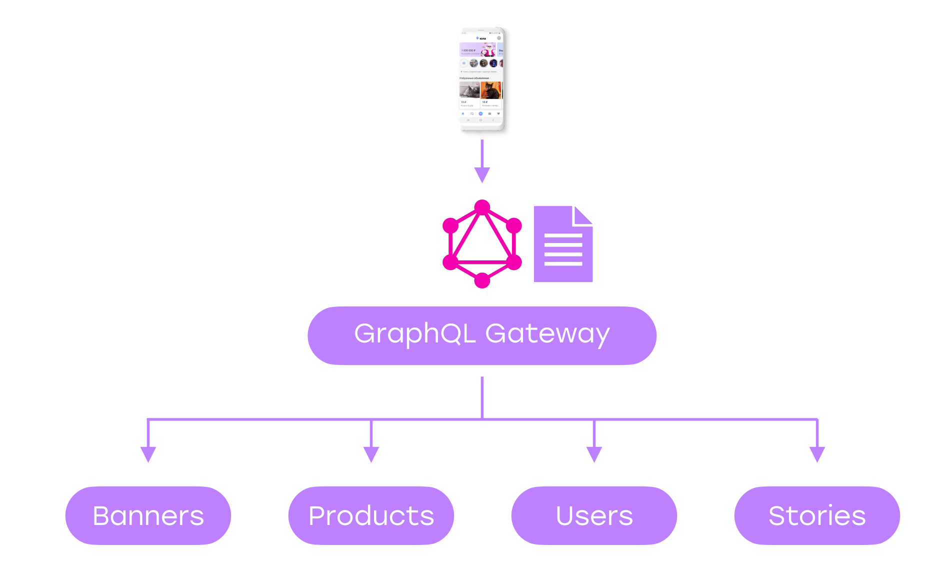
- Одна точка входа для клиентов
- Единая схема для клиентов
- Каждая команда отвечает за свою часть схемы

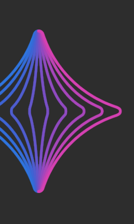




Что мы хотели получить?

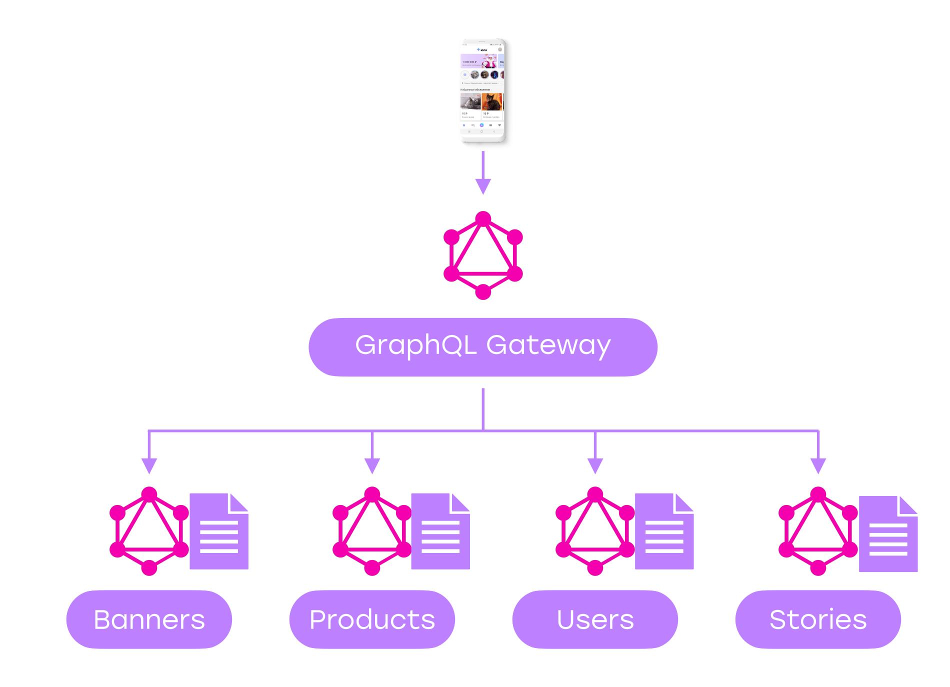
- Одна точка входа для клиентов
- Единая схема для клиентов
- Каждая команда отвечает за свою часть схемы
- Любые изменения схемы «на лету»



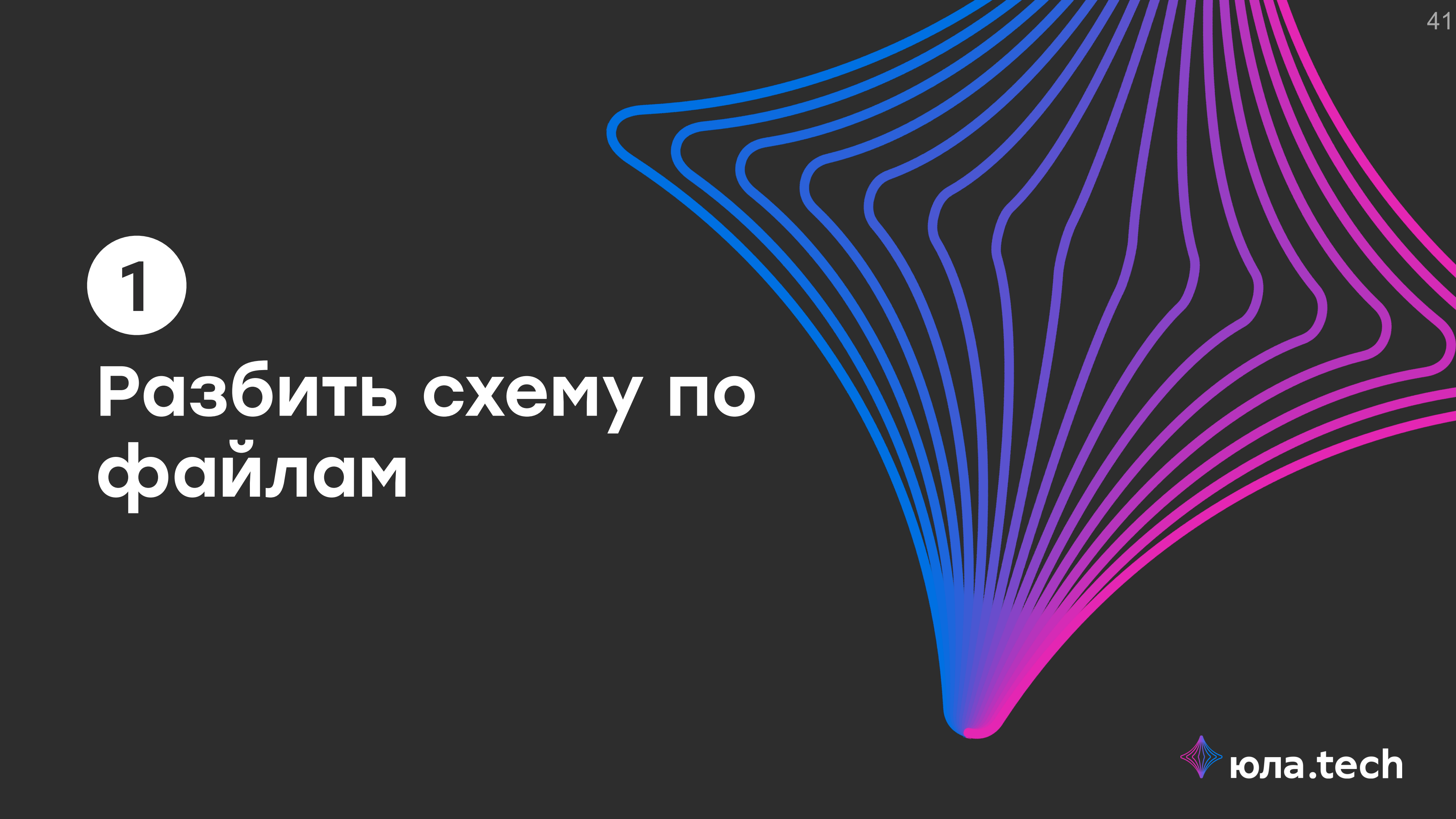


Что мы хотели получить?

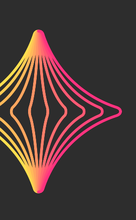
- Одна точка входа для клиентов
- Единая схема для клиентов
- Каждая команда отвечает за свою часть схемы
- Любые изменения схемы «на лету»



ИГОРЬ!



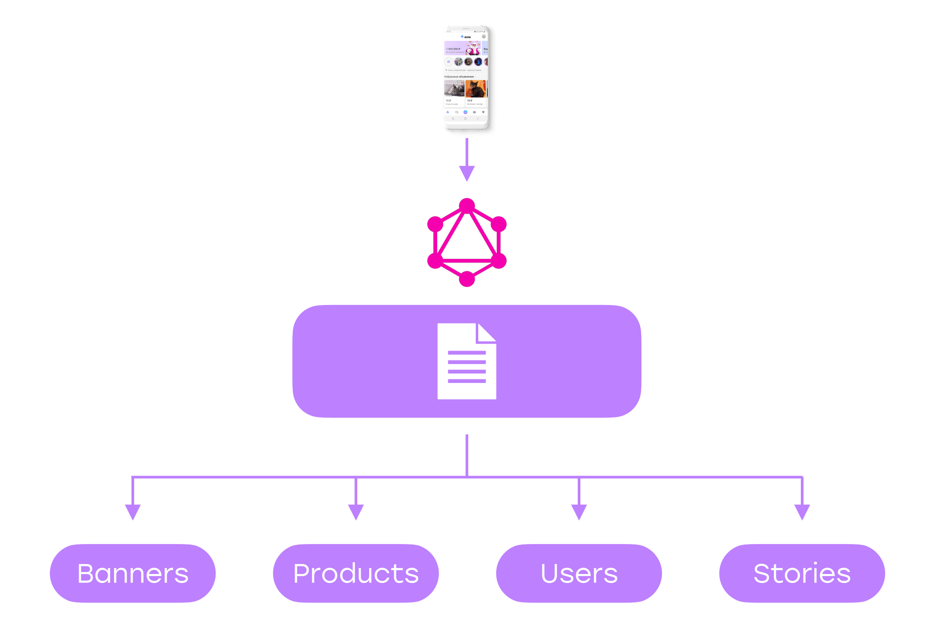
41
1
Разбить схему по файлам
юла.tech



Разбить схему по файлам

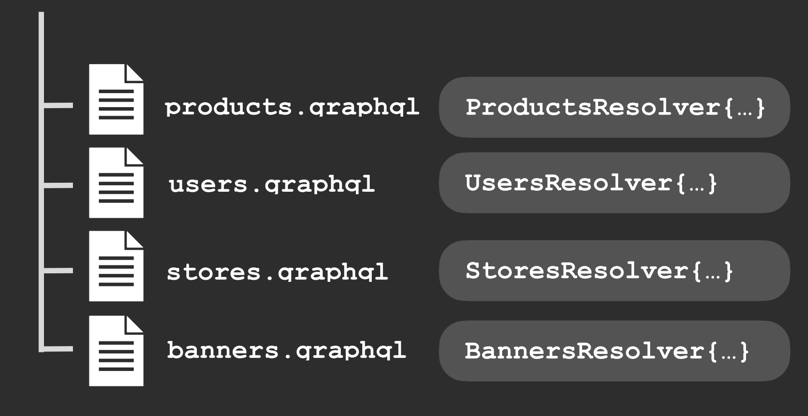
schema.graphql

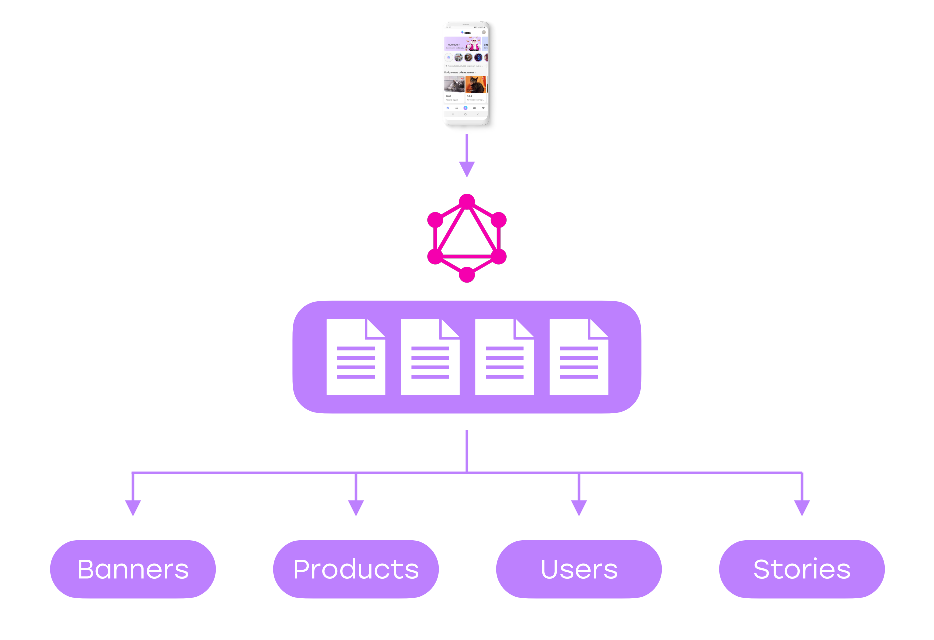
```
SchemaResolver {
  products(…) []Products
  user(id) User
  banners() []Banners
  store(id) Store
}
```





Разбить схему по файлам

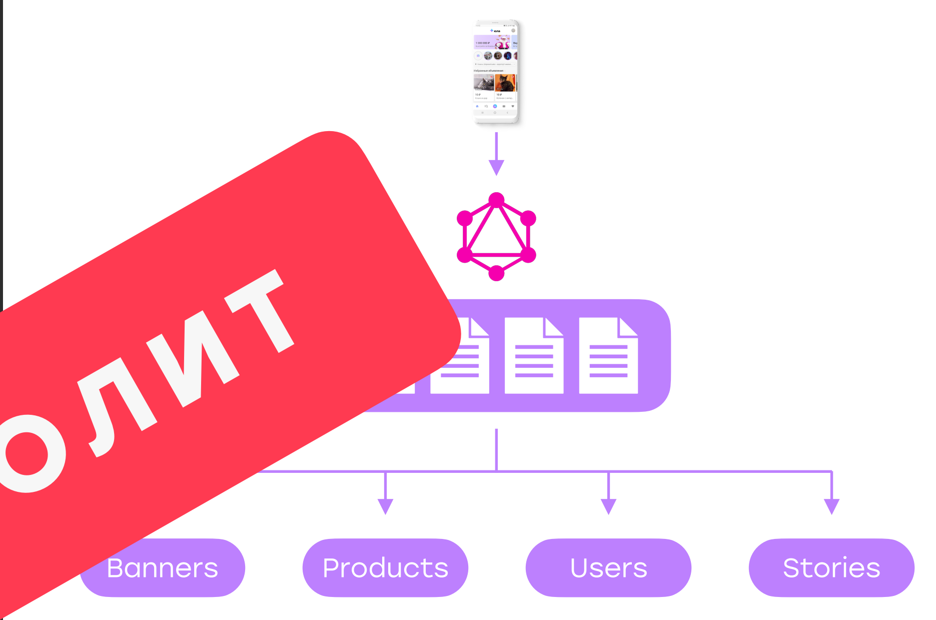






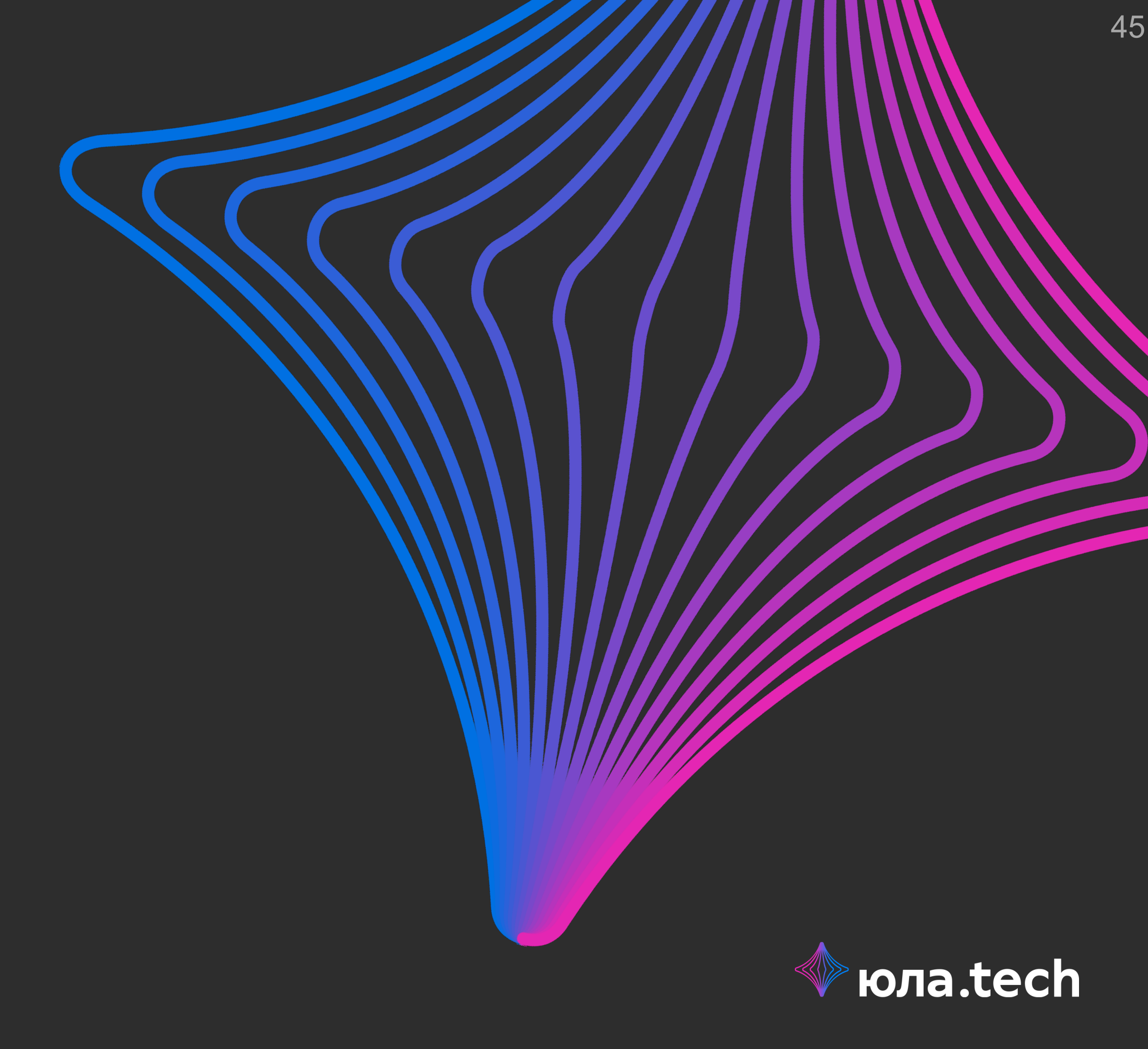
Разбить схему по файлам

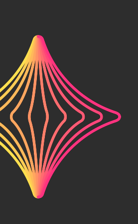




2

GraphQL Federation





GraphQL Federation

Federation

Banners

Products

Users

Stories



GraphQL Federation

- Объединяет схемы

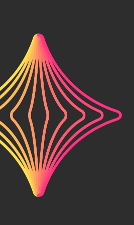
Federation

Banners

Products

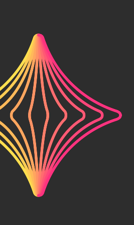
Users

Stories



GraphQL Federation

- Объединяет схемы
- Позволяет расширять типы



GraphQL Federation

- Объединяет схемы
- Позволяет расширять типы
- Валидирует схему



GraphQL Federation

- Объединяет схемы
- Позволяет расширять типы
- Валидирует схем

APPROVE

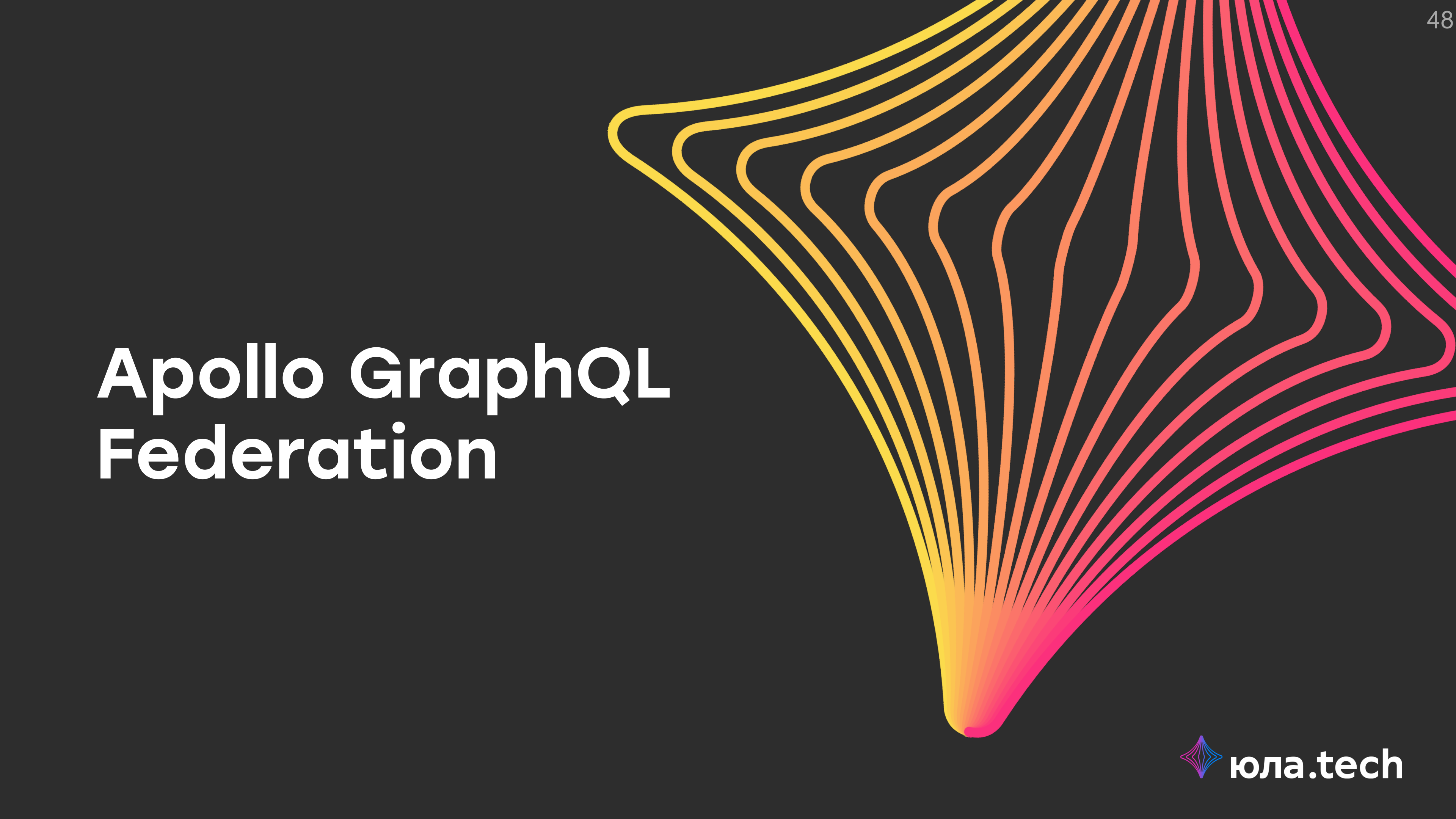
deration

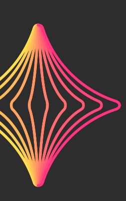
Banners

Products

Users

Stories

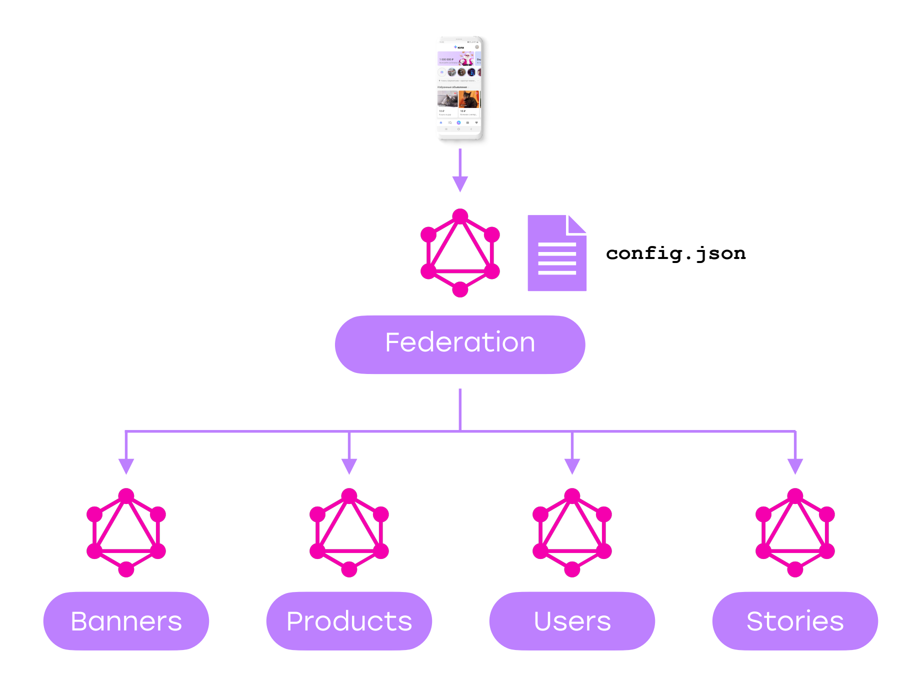

Apollo GraphQL
Federation
юла.tech

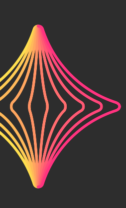


Статичная конфигурация

config.json

```
serviceList: [
  {name: "products", url: "http://
products-graphql/"},
  {name: "users", url: "http://users-
graphql/"},
  {name: "banners", url: "http://banners-
graphql/"},
]
```



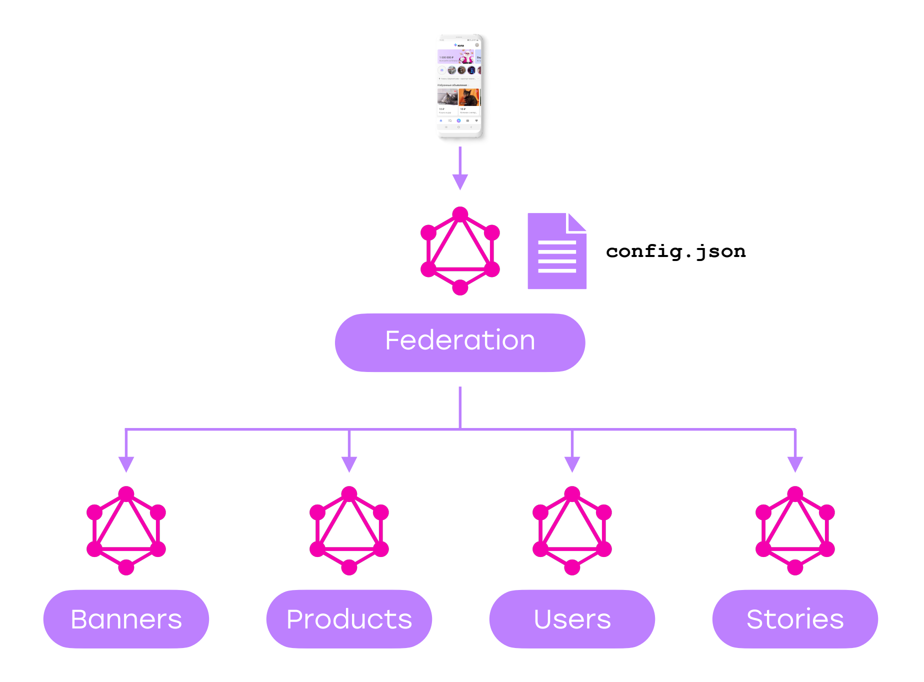


Статичная конфигурация

config.json

```
serviceList: [
  {name: "products", url: "http://products-graphql/"},
  {name: "users", url: "http://users-graphql/"},
  {name: "banners", url: "http://banners-graphql/"},
]
```

- Вы не боитесь конфликтов схем



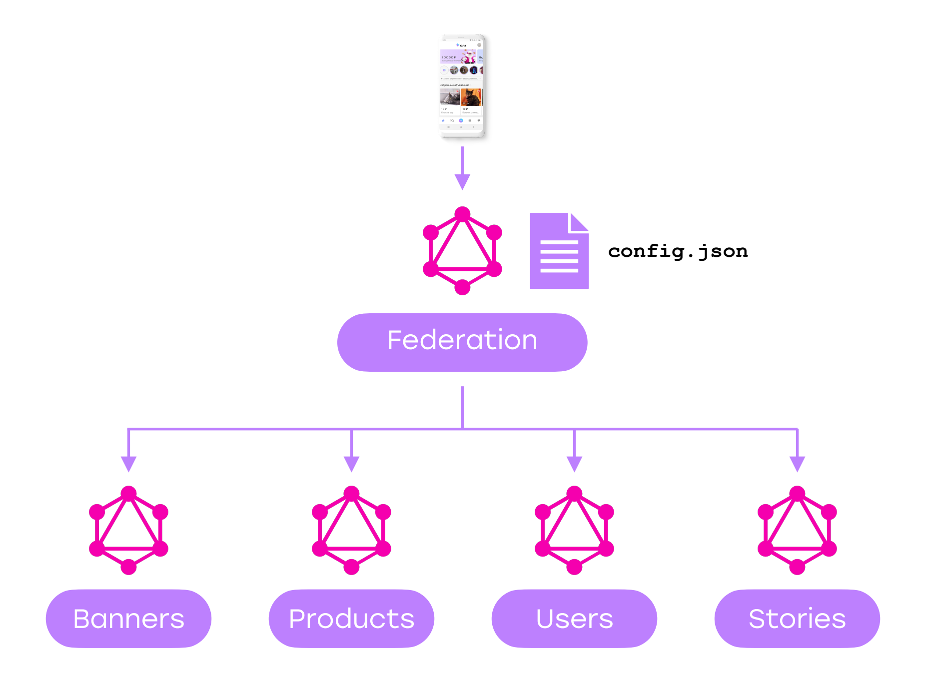


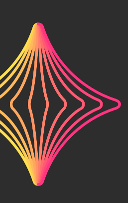
Статичная конфигурация

config.json

```
serviceList: [
  {name: "products", url: "http://
products-graphql/"},
  {name: "users", url: "http://users-
graphql/"},
  {name: "banners", url: "http://banners-
graphql/"},
]
```

- Вы не боитесь конфликтов схем
- Вы не распиливаете монолитную схему



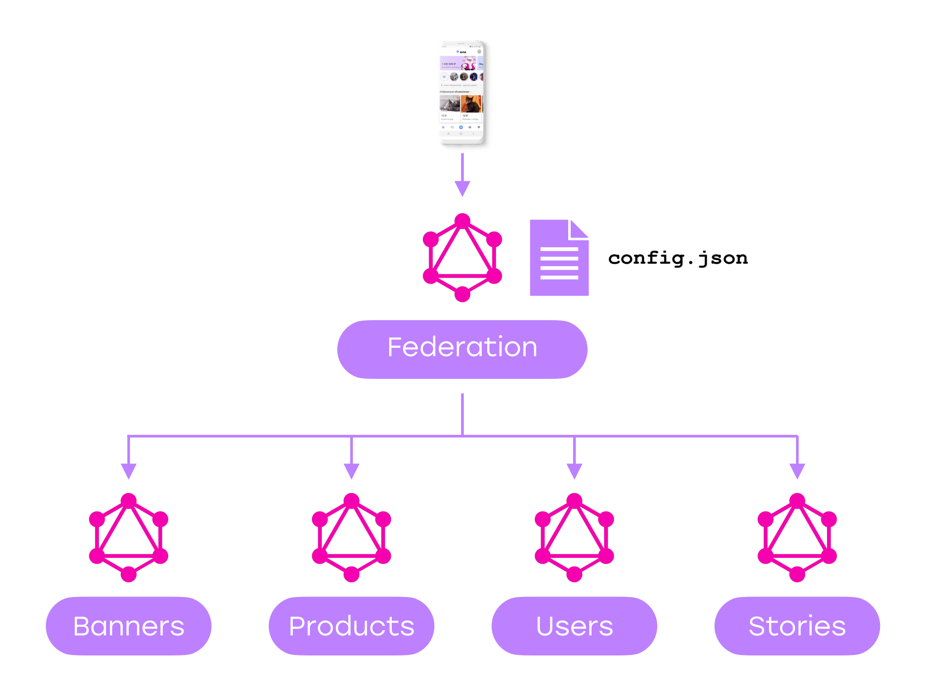


Статичная конфигурация

config.json

```
serviceList: [
  {name: "products", url: "http://
products-graphql/"},
  {name: "users", url: "http://users-
graphql/"},
  {name: "banners", url: "http://banners-
graphql/"},
]
```

- Вы не боитесь конфликтов схем
- Вы не распиливаете монолитную схему
- Кратковременный даунтайм - не проблема

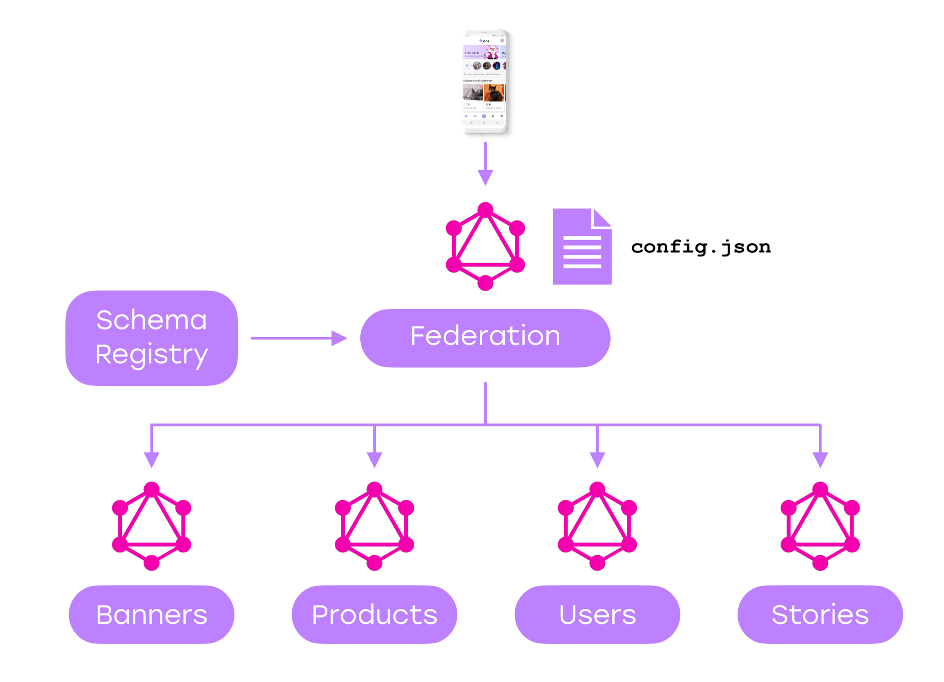




Managed Federation

config.json

```
{
  registryURL: "http://schema-registry/"
}
```



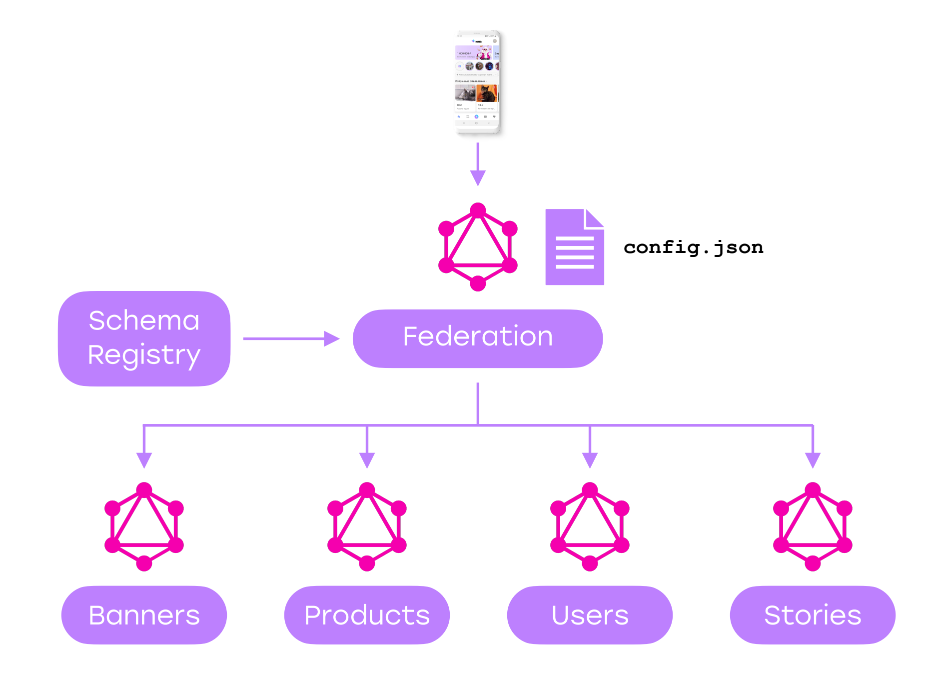


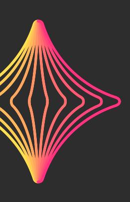
Managed Federation

config.json

```
{
  registryURL: "http://schema-registry/"
}
```

- Нужен schema registry



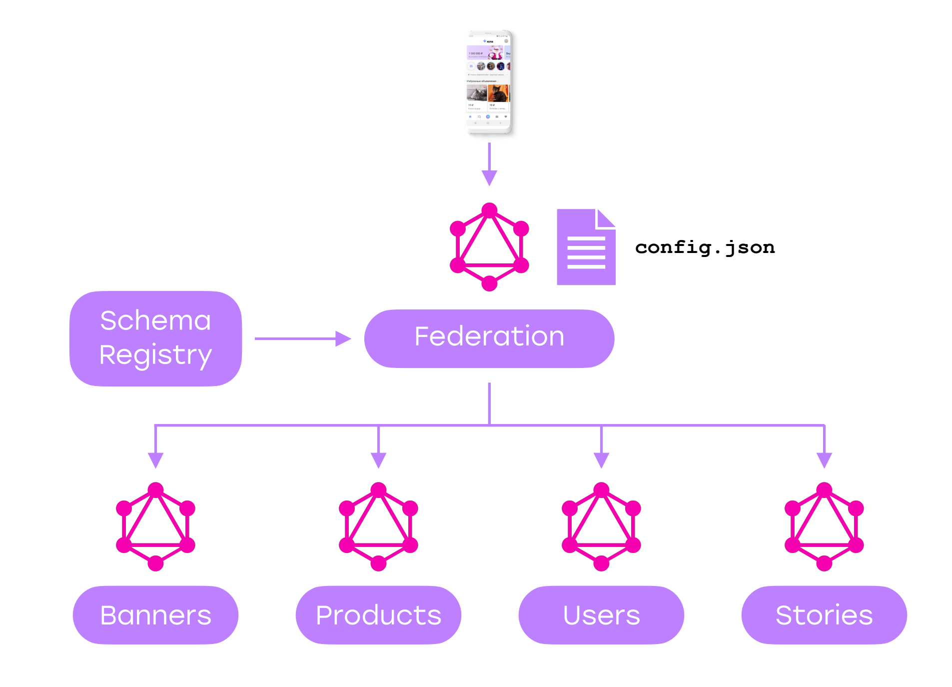


Managed Federation

config.json

```
{
  registryURL: "http://schema-registry/"
}
```

- Нужен schema registry
- Добавление схем в Schema Registry



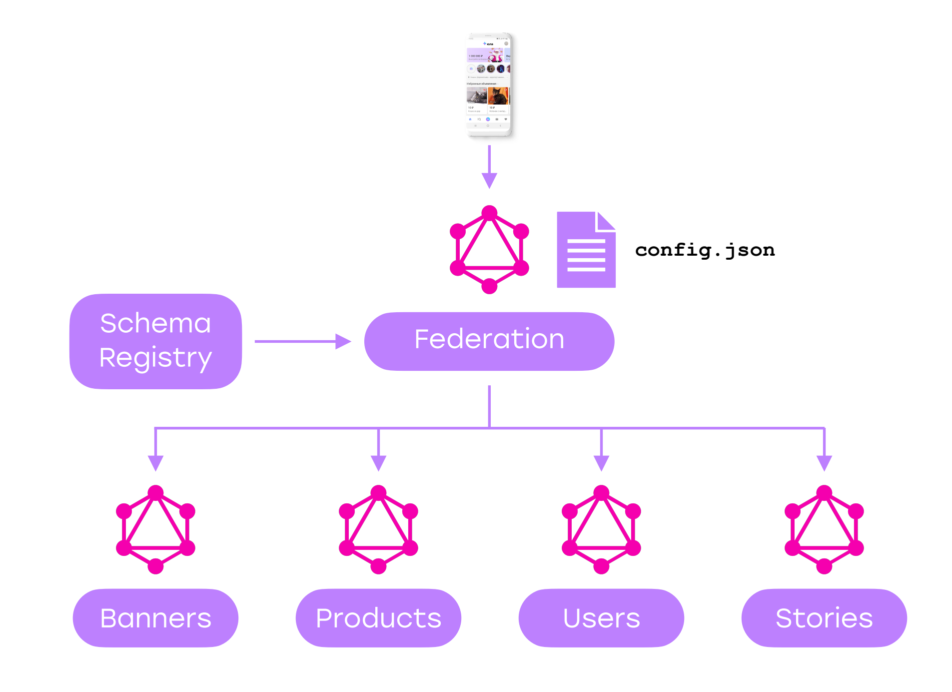


Managed Federation

config.json

```
{
  registryURL: "http://schema-registry/"
}
```

- Нужен schema registry
- Добавление схем в Schema Registry
- Получение схем из Schema Registry



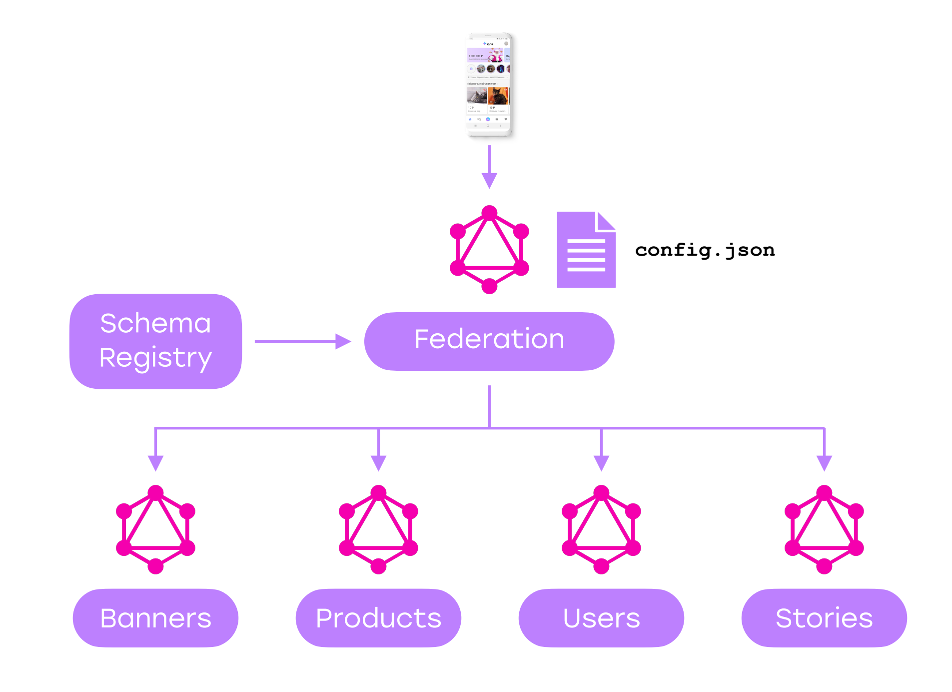


Managed Federation

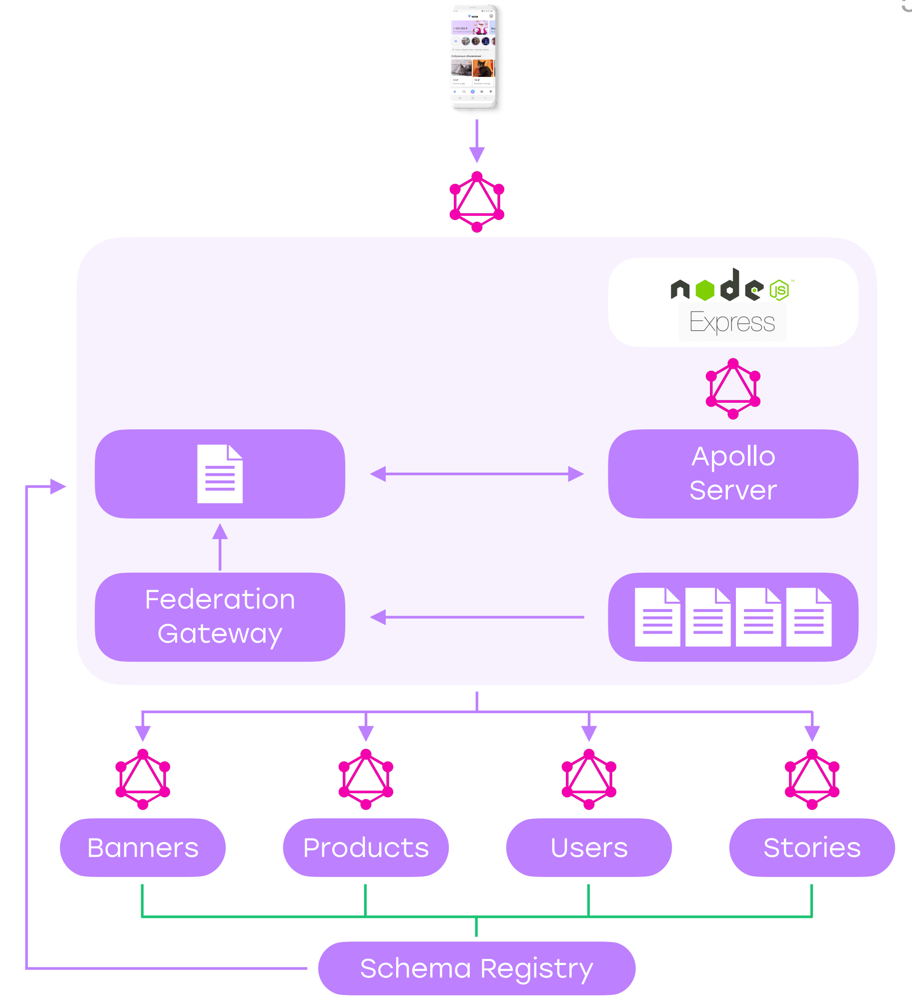
config.json

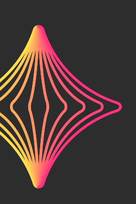
```
{
  registryURL: "http://schema-registry/"
}
```

- Нужен schema registry
- Добавление схем в Schema Registry
- Получение схем из Schema Registry

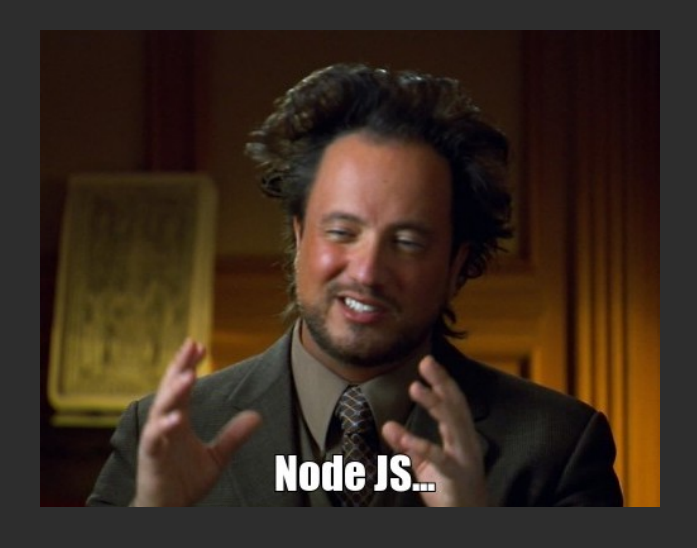


Что такое федерация



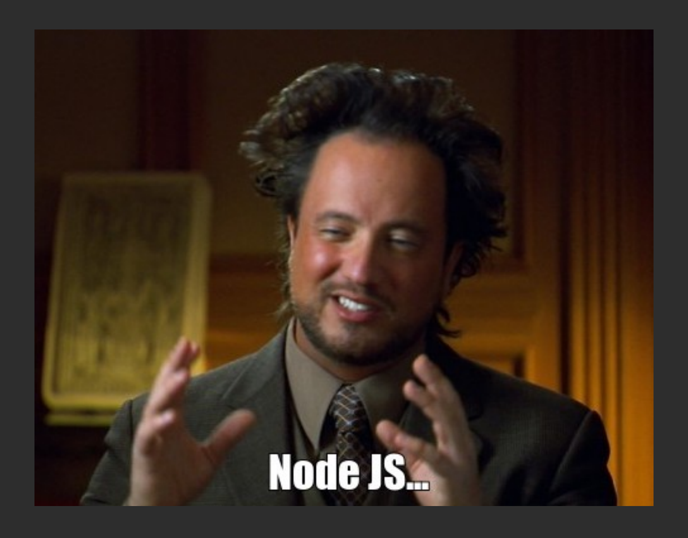


Что такое федерация





Что такое федерация



- Валидность актуальной схемы

Express

Apollo Server

Federation Gateway

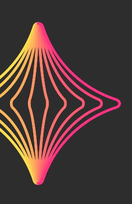
Banners

Products

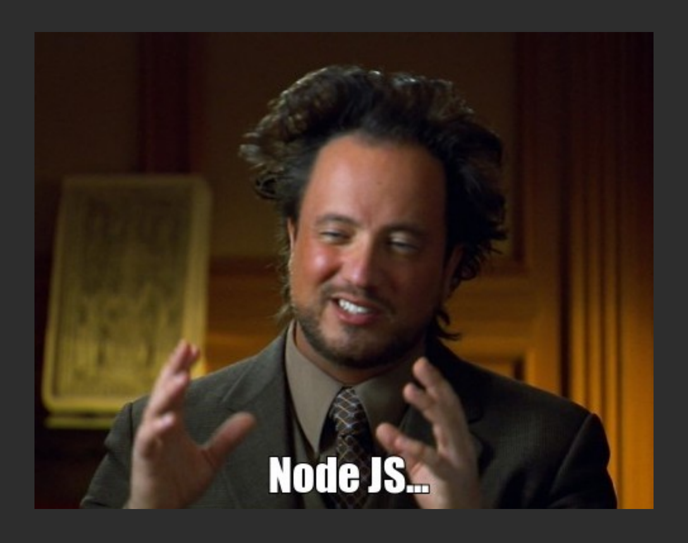
Users

Stories

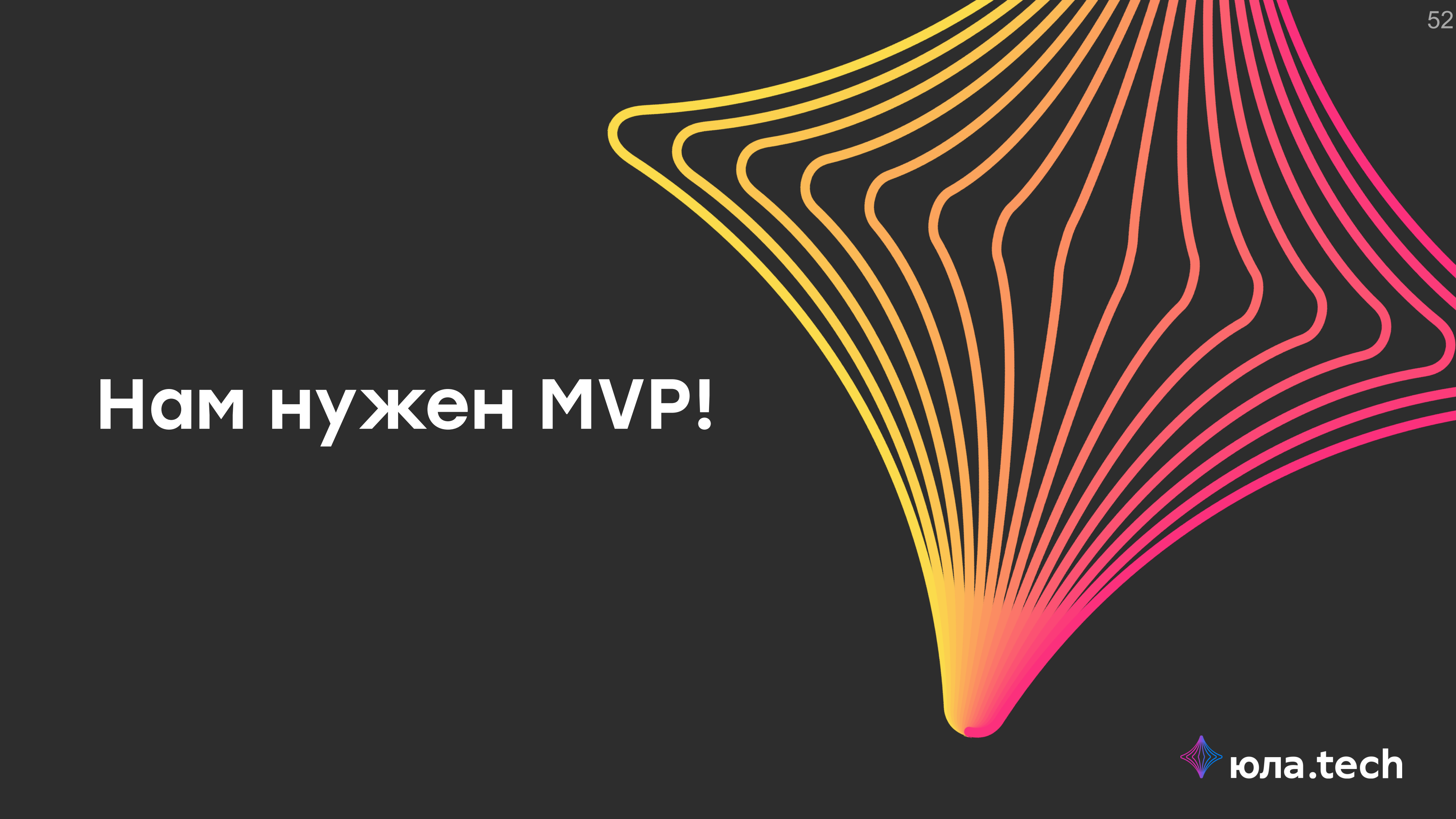
Schema Registry



Что такое федерация

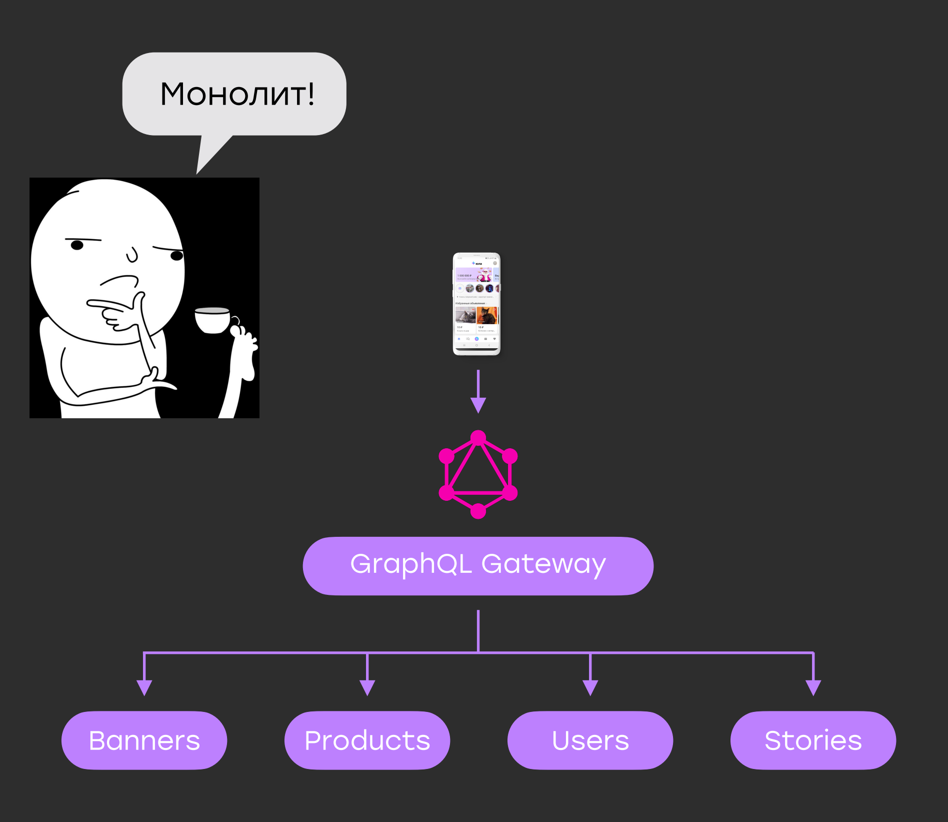


- Валидность актуальной схемы
- Обновляется без передеплоя


52
Нам нужен MVP!
юла.tech


MVP Federation

MVP Federation



MVP Federation

Монолит!

GraphQL Gateway

Banners
Products
Users
Stories

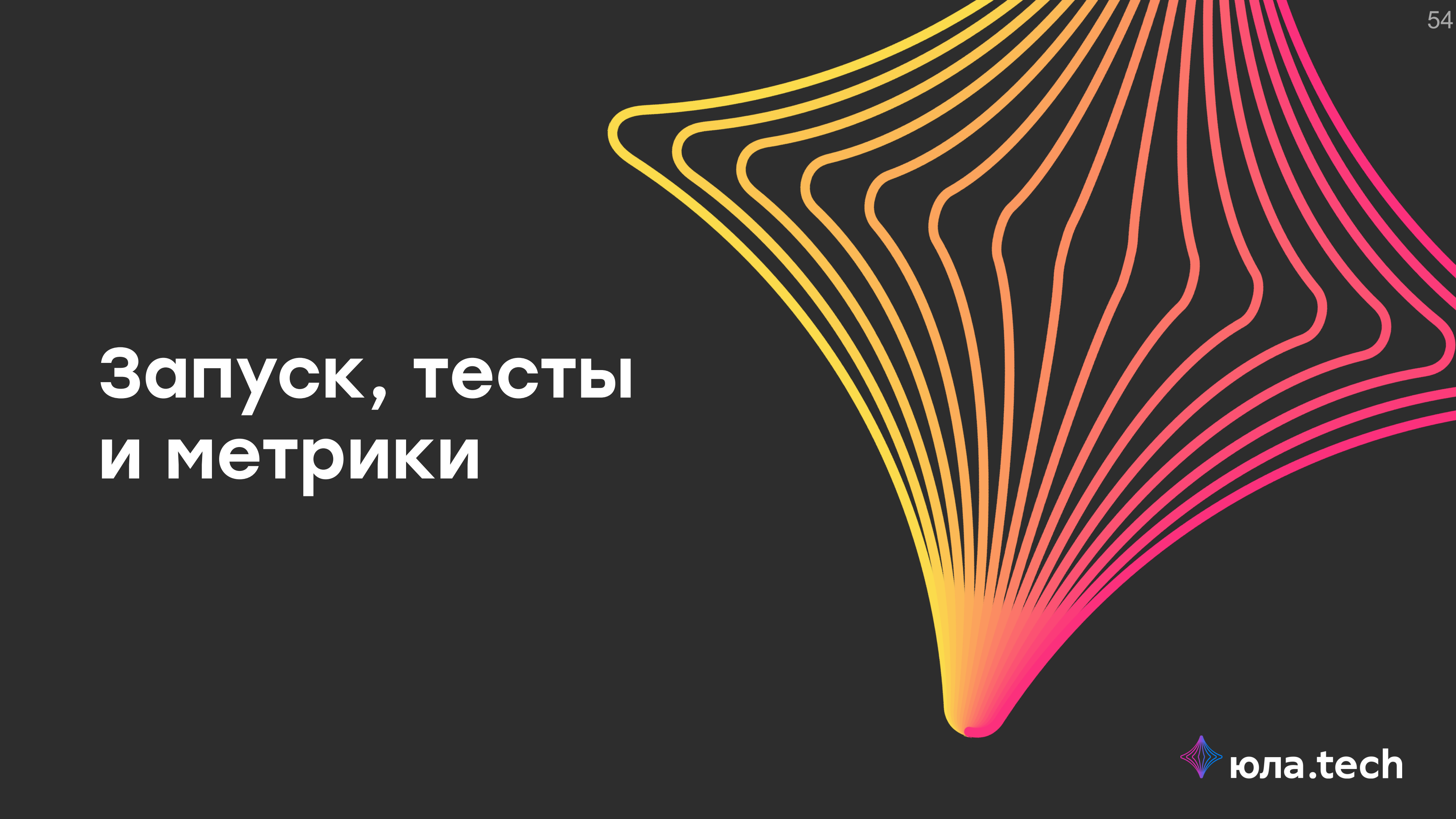
Уже не монолит!

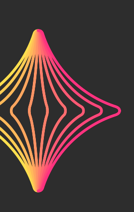
Schema Registry

Federation

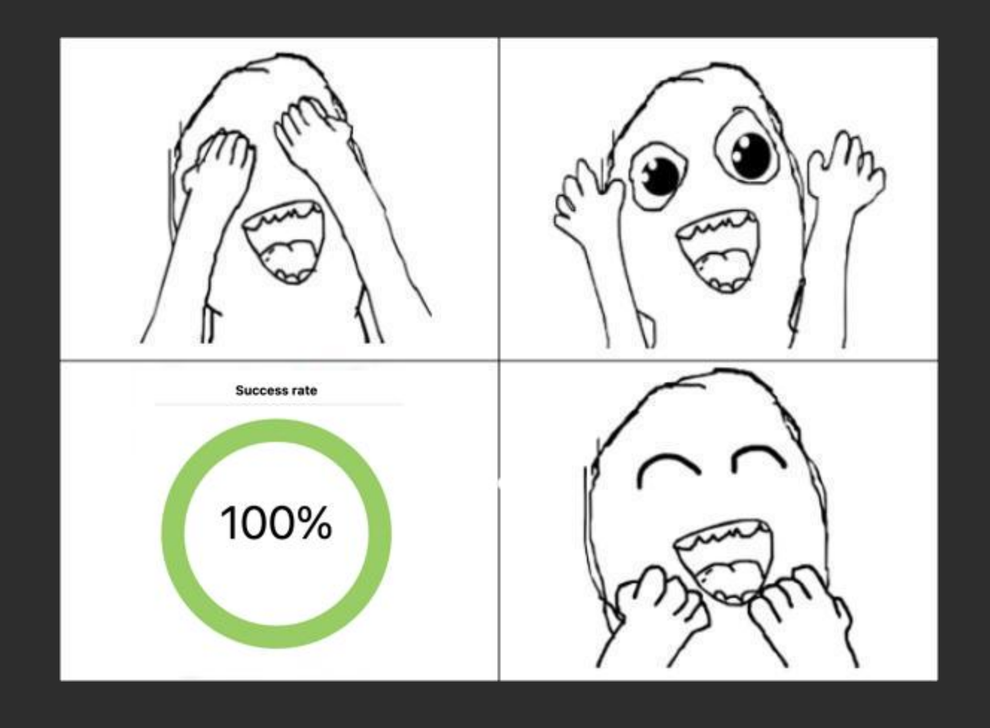
GraphQL Gateway

Banners
Products
Users
Stories

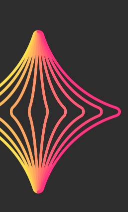

54
Запуск, тесты
и метрики
юла.tech



Тестирование





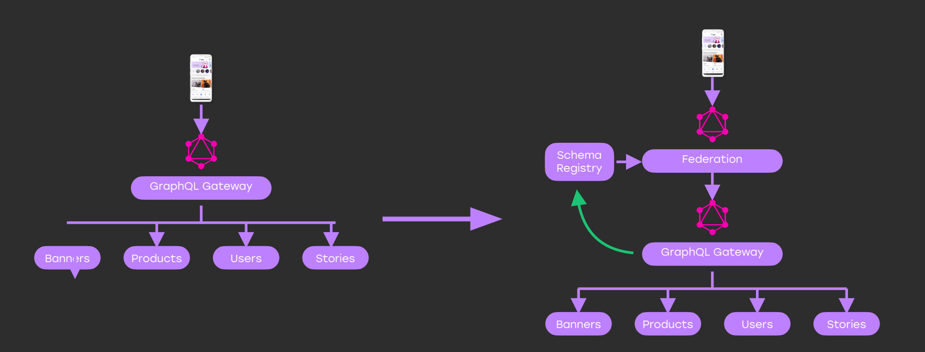


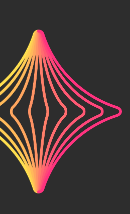
Что по скорости?





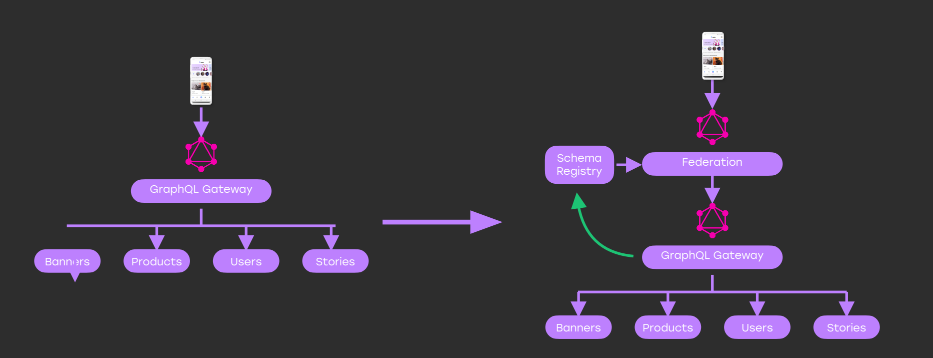
Как запускали

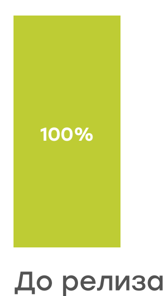


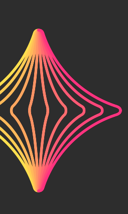


Как запускали

- Замерили скорость до релиза

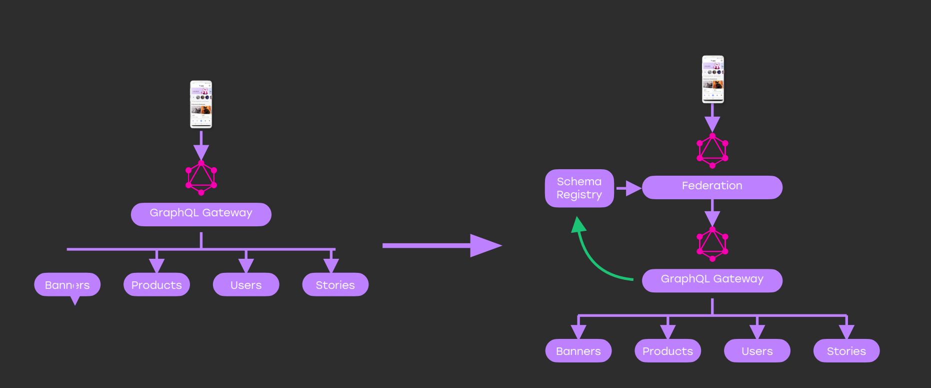




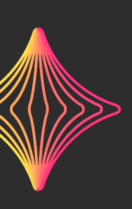


Как запускали

- Замерили скорость до релиза
- Запуск MVP

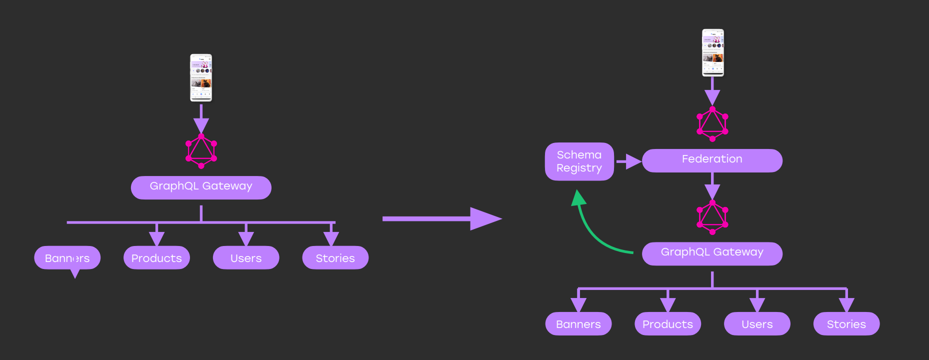


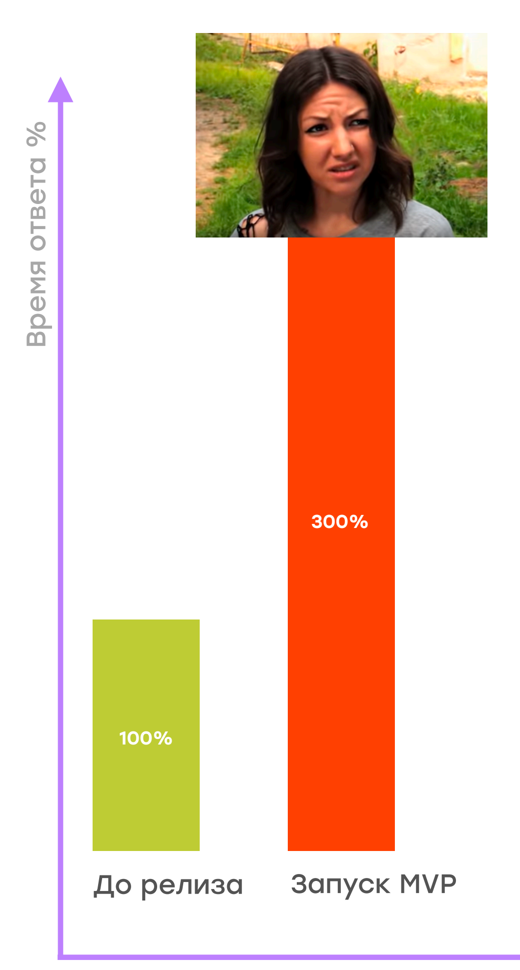


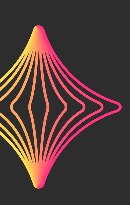


Как запускали

- Замерили скорость до релиза
- Запуск MVP

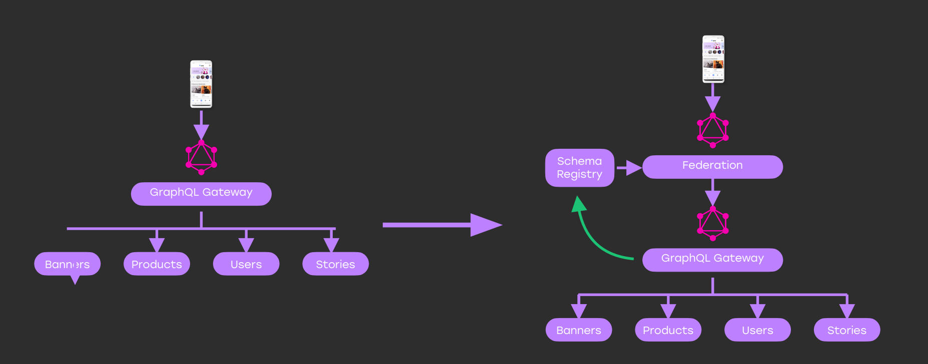






Как запускали

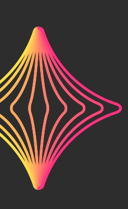
- Замерили скорость до релиза
- Запуск MVP
- Добавили нод



Время ответа %

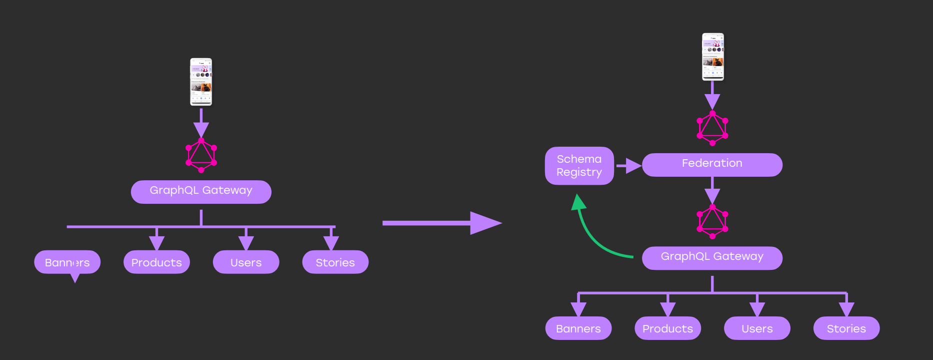
До релиза	Запуск MVP	Добавили нод
100%	300%	160%

Релизы



Как запускали

- Замерили скорость до релиза
- Запуск MVP
- Добавили нод
- Фикс



Время ответа %

100%

300%

160%

105%

Async
Await
Async
Await
Async
Await

До релиза

Запуск MVP

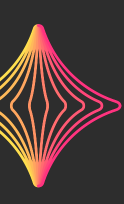
Добавили нод

Фикс

Релизы

А теперь распиливаем схему





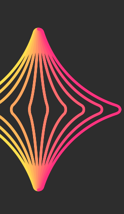
Распиливаем схему на микросервисы



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол

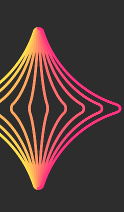
Schema Registry
Federation
GraphQL Gateway
Banners
Products
Users
Stories



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол

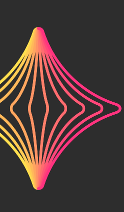
Schema Registry
Federation
GraphQL Gateway
Banners
Products
Users
Stories



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол
- Настроить последовательный push схемы в SR

Schema Registry
Federation
GraphQL Gateway
Banners
Products
Users
Stories



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол
- Настроить последовательный push схемы в SR

Schema Registry

Federation

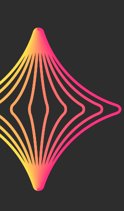
GraphQL Gateway

Banners

Products

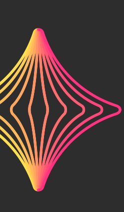
Users

Stories



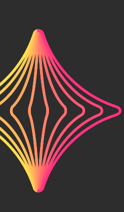
Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол
- Настроить последовательный push схемы в SR



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол
- Настроить последовательный push схемы в SR



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол
- Настроить последовательный push схемы в SR
- Убрать GraphQL Gateway

Schema Registry

Federation

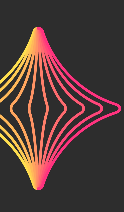
GraphQL Gateway

Banners

Products

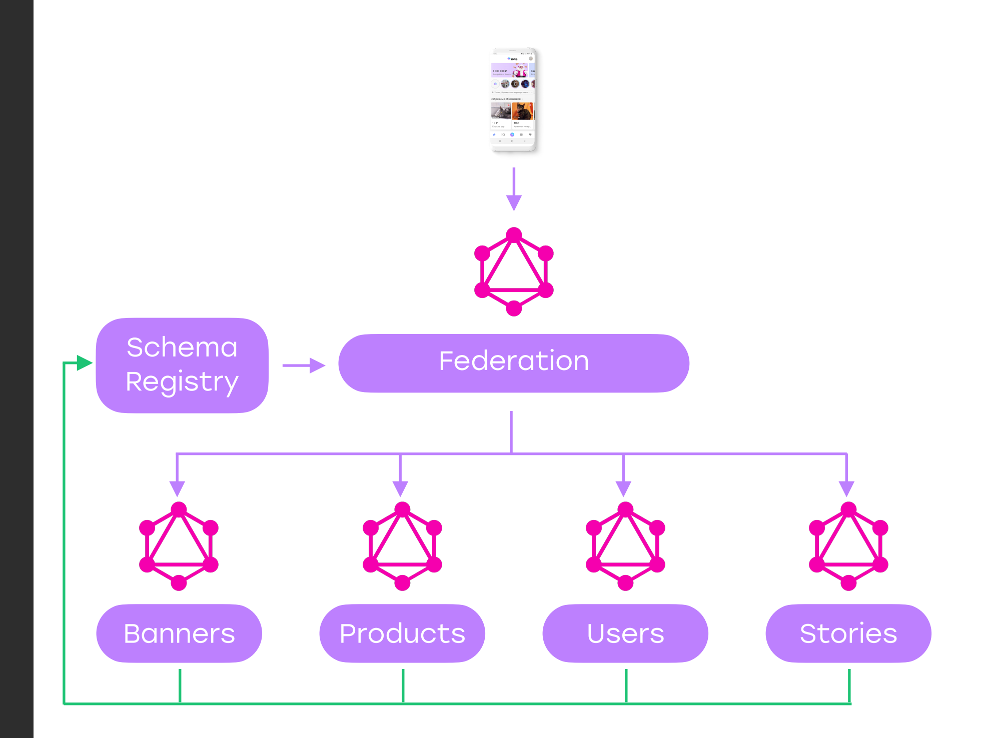
Users

Stories



Распиливаем схему на микросервисы

- Добавить целевым сервисам GraphQL протокол
- Настроить последовательный push схемы в SR
- Убрать GraphQL Gateway





GraphQL Gateway

```
type Query {
    products(limit: Int): [Product!]
    product(id: ID!): Product
    user(id: ID!): User
    me: User
    . . .
}

# Базовый пользователь
type User {
    id: ID!
    firstName: String!
    lastName: String!
    products(limit: Int!): [Product!]
}

# Базовый продукт/объявление
type Product {
    id: ID!
    url: String!
    owner: User!
    name: String!
}
```





GraphQL Gateway

```
type Query {
    products(limit: Int): [Product!]
    product(id: ID!): Product
    user(id: ID!): User
    me: User
    . . .
}

# Базовый пользователь
type User {
    id: ID!
    firstName: String!
    lastName: String!
    products(limit: Int!): [Product!]
}

# Базовый продукт/объявление
type Product {
    id: ID!
    url: String!
    owner: User!
    name: String!
}
```

Products

```
extend type Query {
    products(limit: Int): [Product!]
    product(id: ID!): Product
    . . .
}

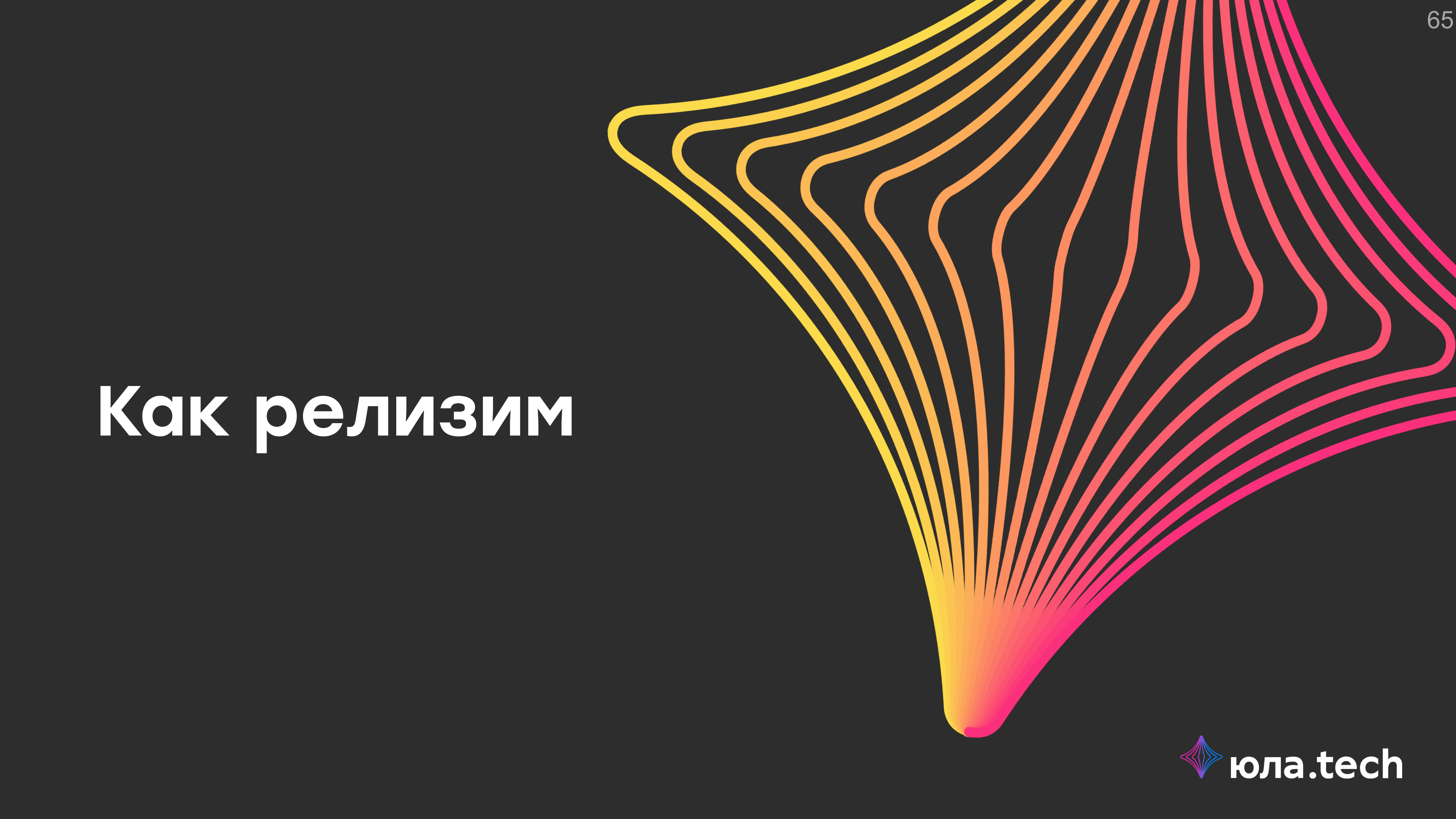
# Базовый продукт/объявление
type Product @key(fields: "id") {
    id: ID!
    url: String!
    owner: User!
    name: String!
}

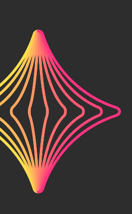
extend User @key(fields: "id") {
    id: ID! @external
    # Добавляем объявления
    products(limit: Int): [Product!]
}
```

Users

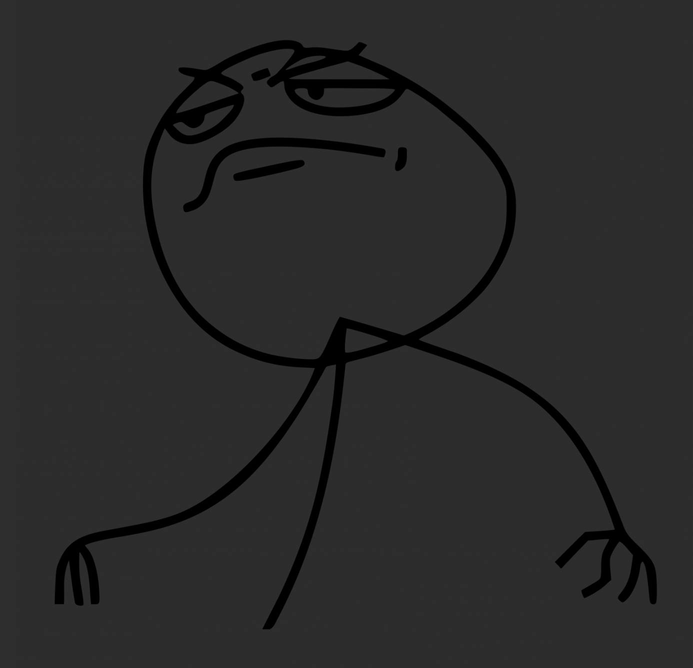
```
extend type Query {
    user(id: ID!): User
    me: User
}

# Базовый пользователь
type User @key(fields: "id") {
    id: ID!
    firstName: String!
    lastName: String!
}
```

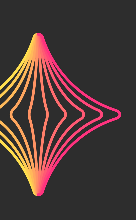

65
Как релизим
юла.tech



Вооружились Schema Registry

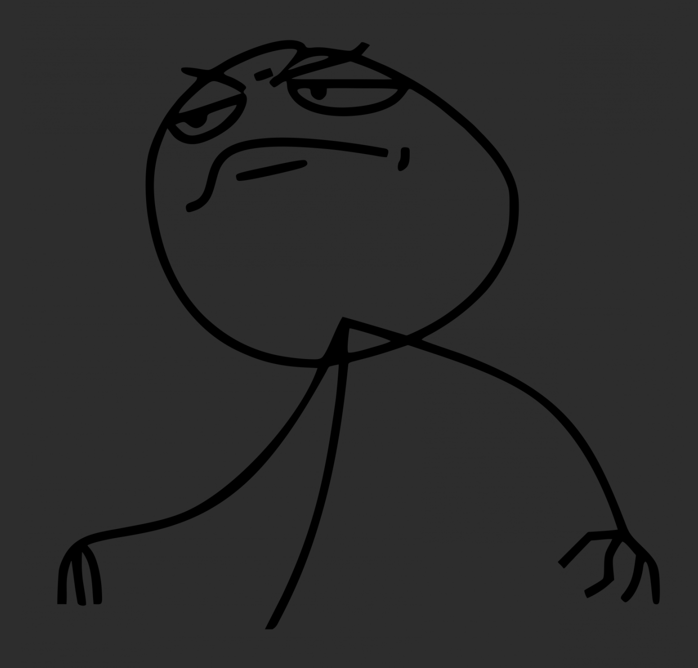




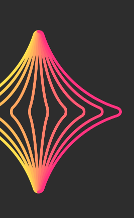


Вооружились Schema Registry

- Сохранять новые схемы

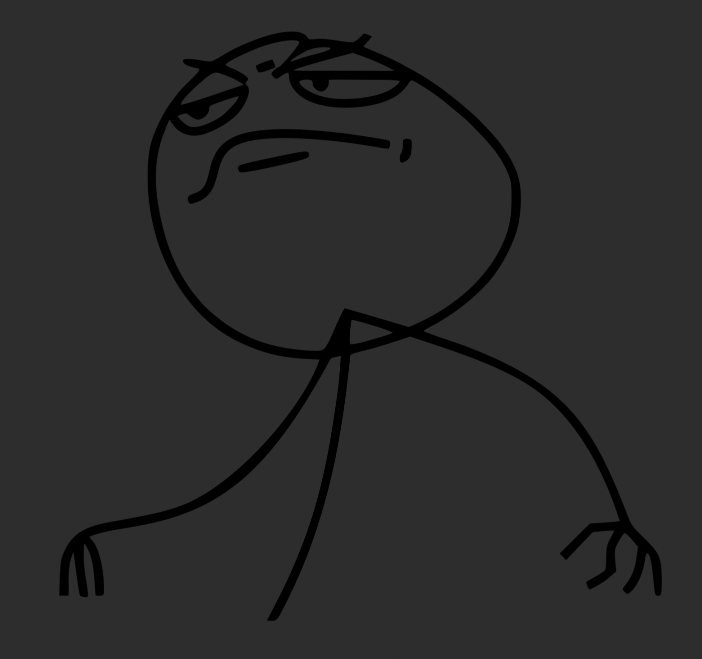




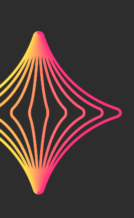


Вооружились Schema Registry

- Сохранять новые схемы
- Создавать их версии

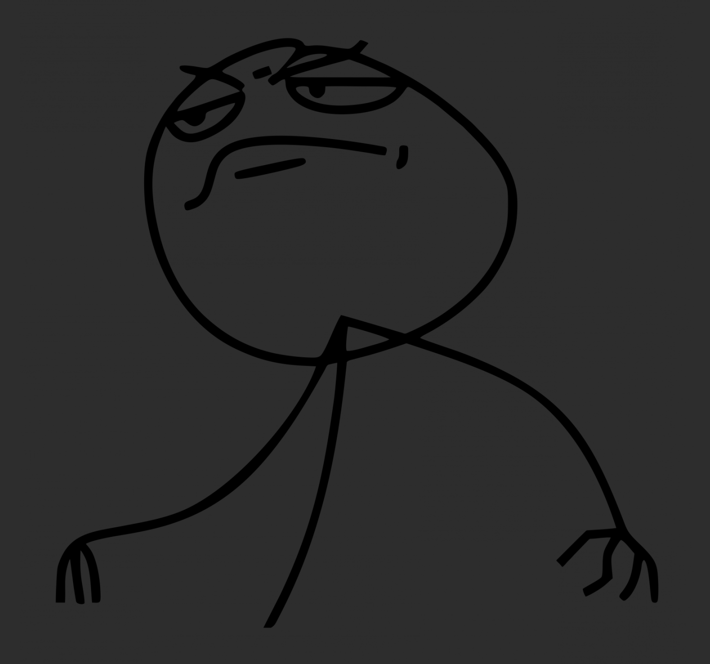




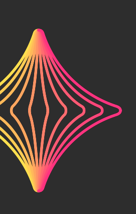


Вооружились Schema Registry

- Сохранять новые схемы
- Создавать их версии
- Валидировать





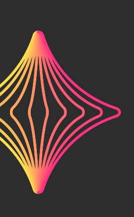


Вооружились Schema Registry

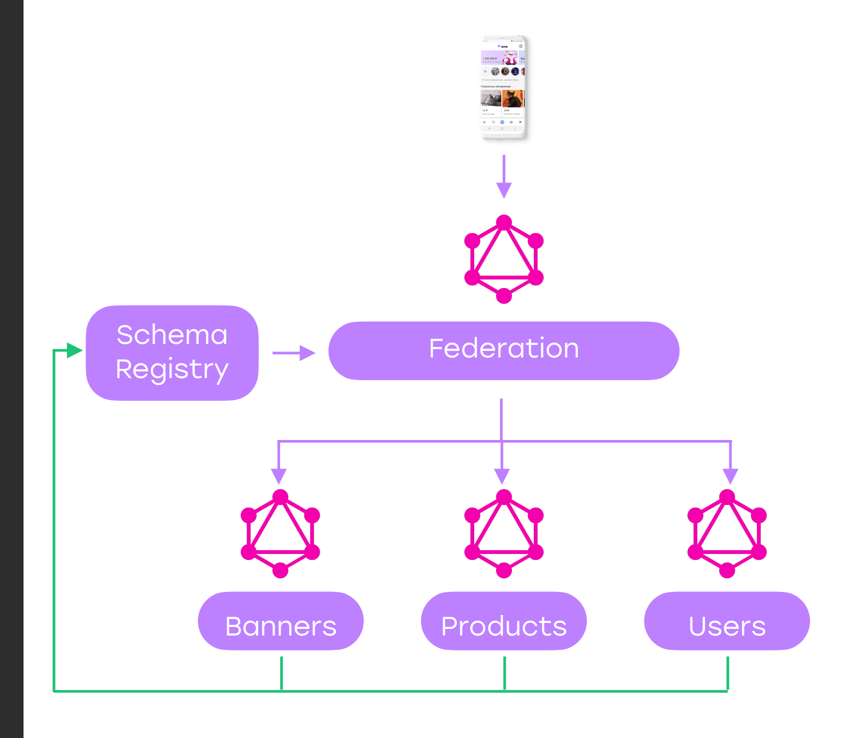
- Сохранять новые схемы
- Создавать их версии
- Валидировать







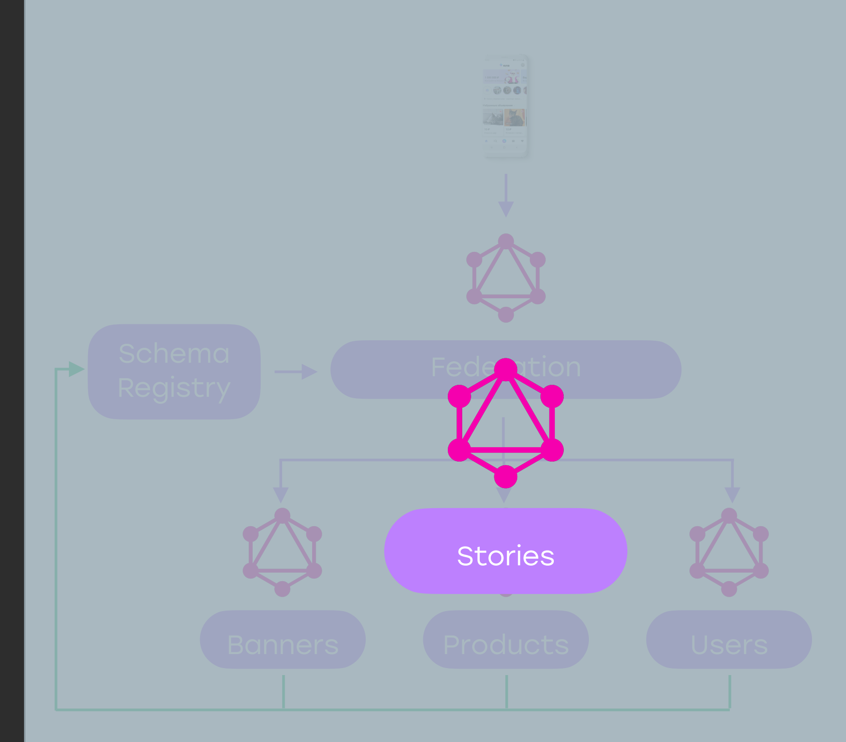
Релиз нового сервиса





Релиз нового сервиса

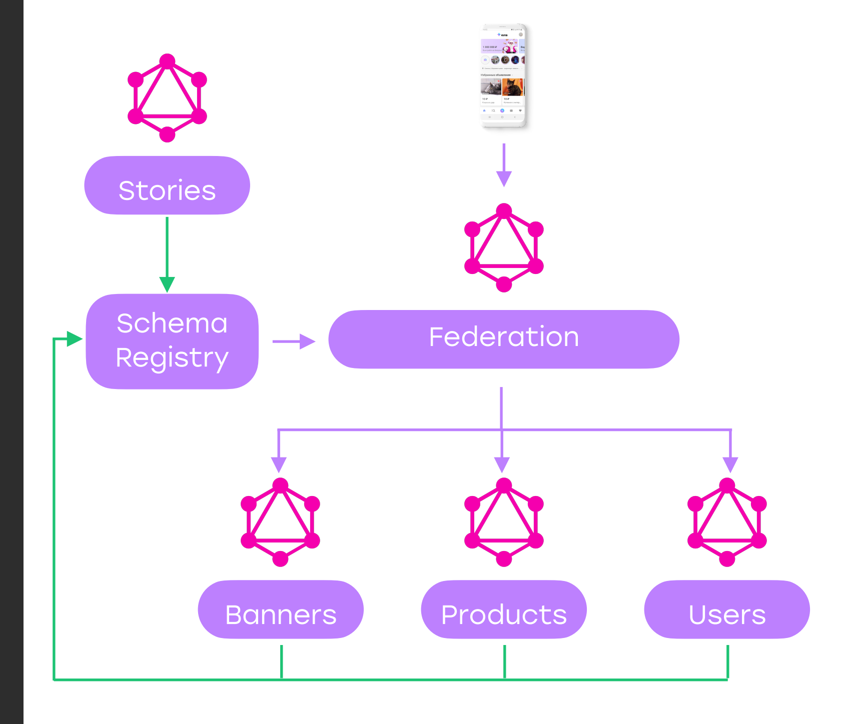
- Создание сервиса с GraphQL интерфейсом





Релиз нового сервиса

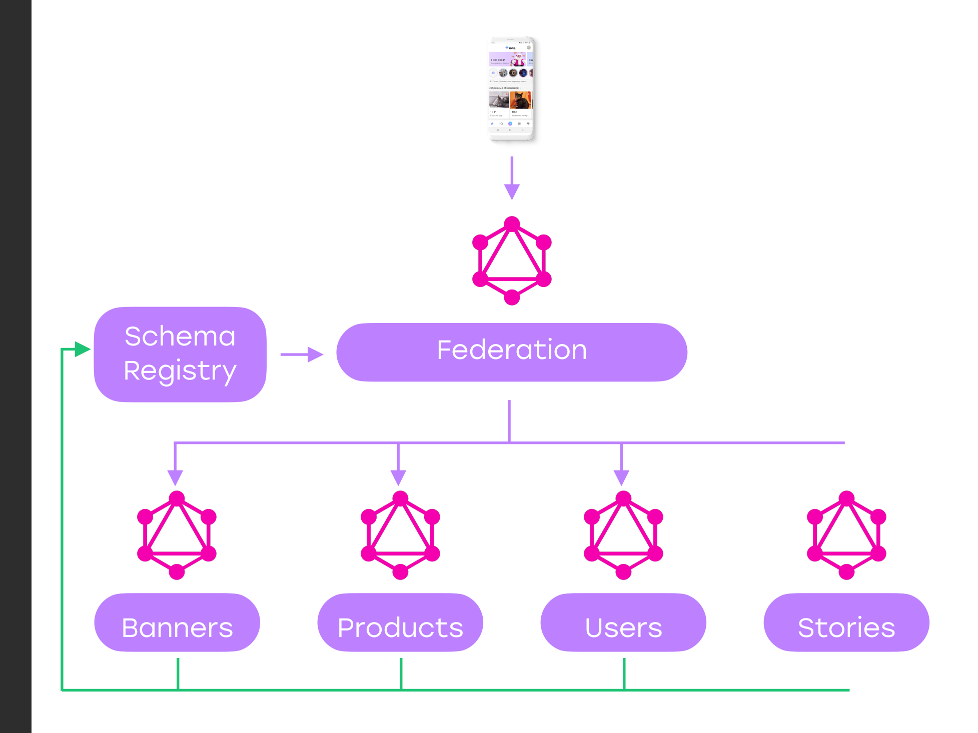
- Создание сервиса с GraphQL интерфейсом
- Валидация graphQL схемы в Schema Registry





Релиз нового сервиса

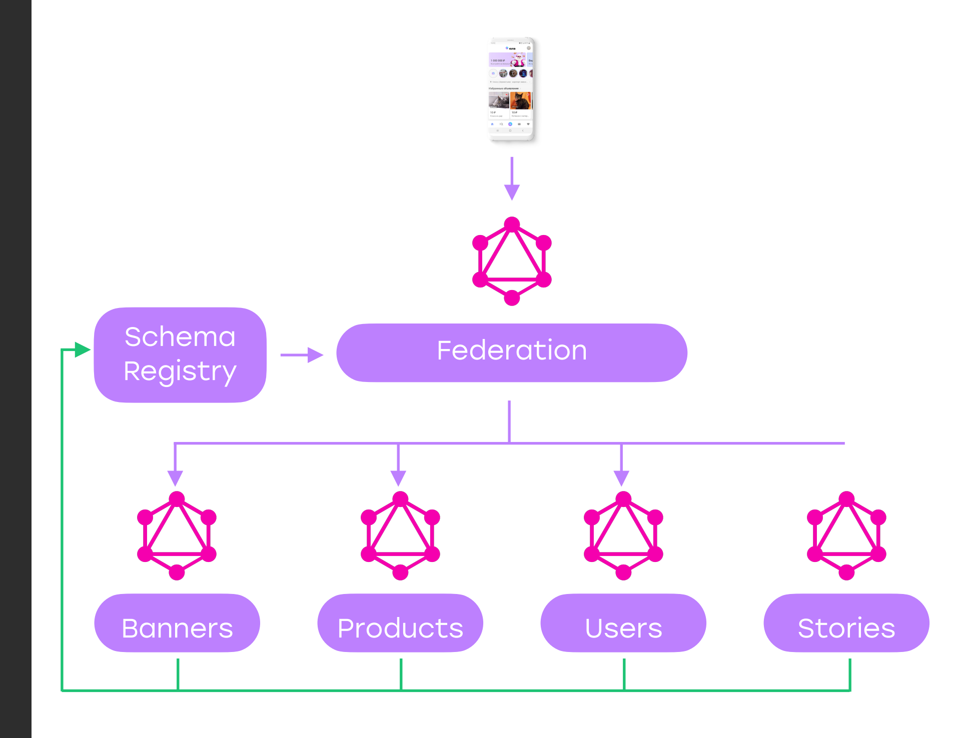
- Создание сервиса с GraphQL интерфейсом
- Валидация graphQL схемы в Schema Registry
- Деплой сервиса





Релиз нового сервиса

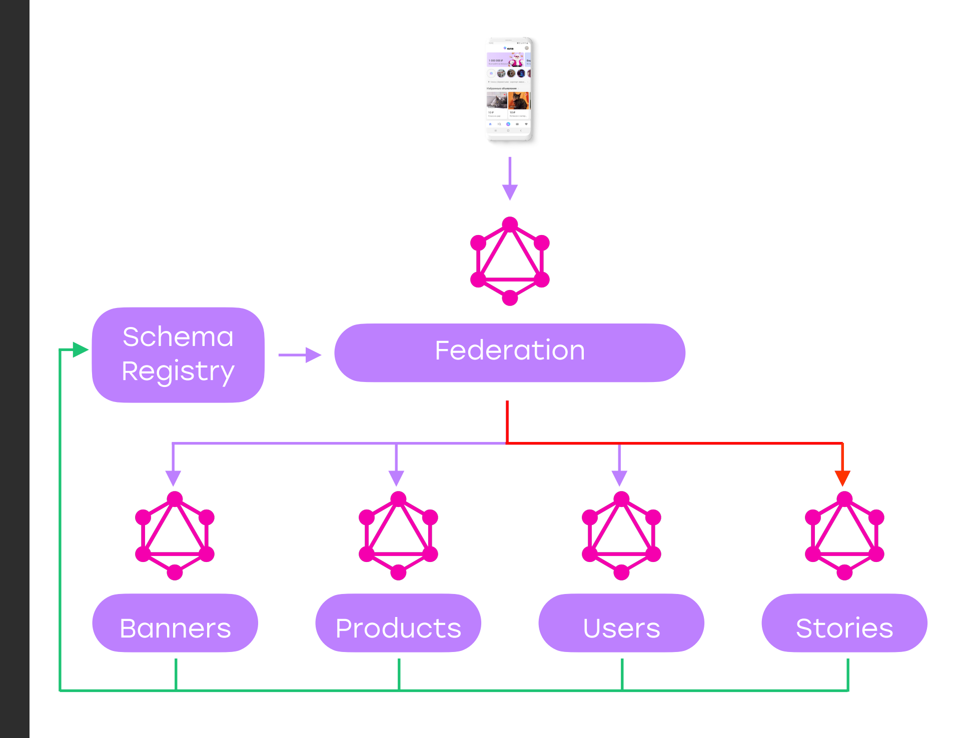
- Создание сервиса с GraphQL интерфейсом
- Валидация graphQL схемы в Schema Registry
- Деплой сервиса
- Добавление схемы в Schema Registry

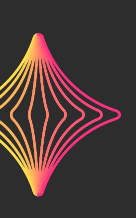




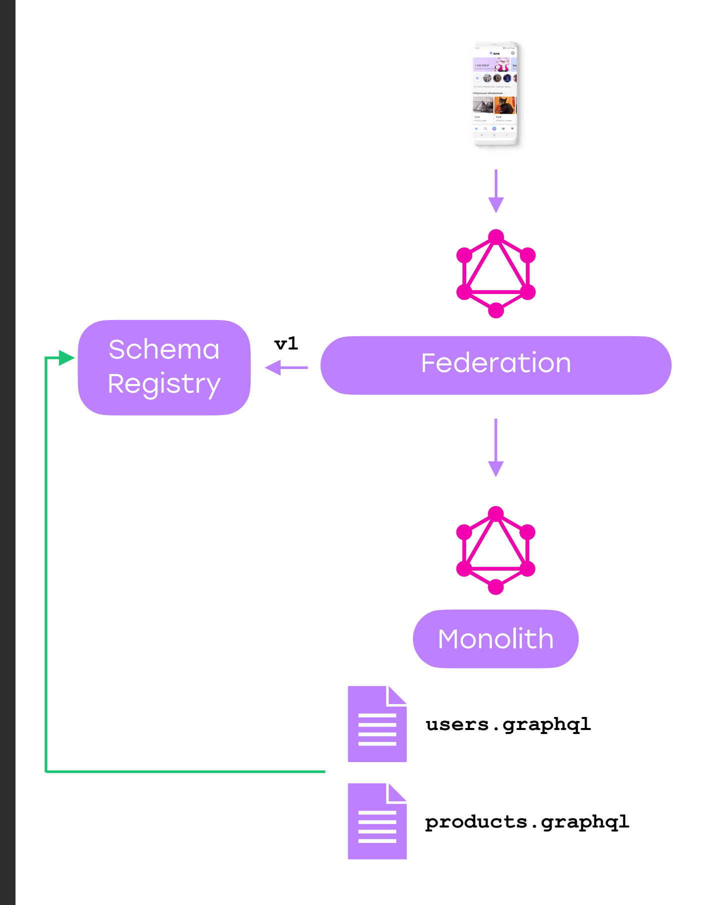
Релиз нового сервиса

- Создание сервиса с GraphQL интерфейсом
- Валидация graphQL схемы в Schema Registry
- Деплой сервиса
- Добавление схемы в Schema Registry
- Получение федерацией новой версии схемы





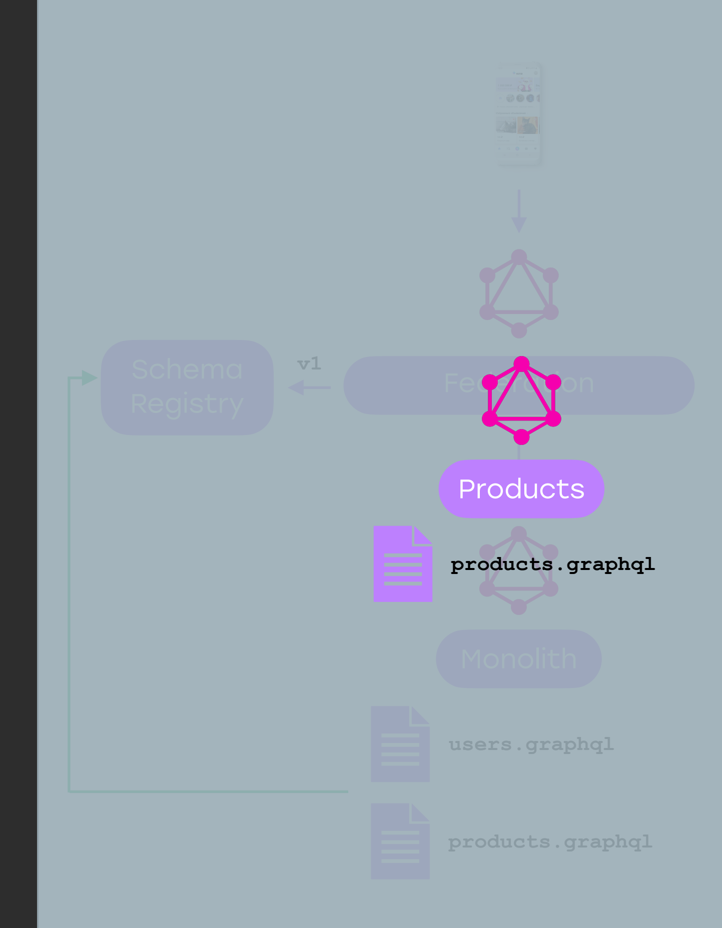
Миграция схемы между сервисами





Миграция схемы между сервисами

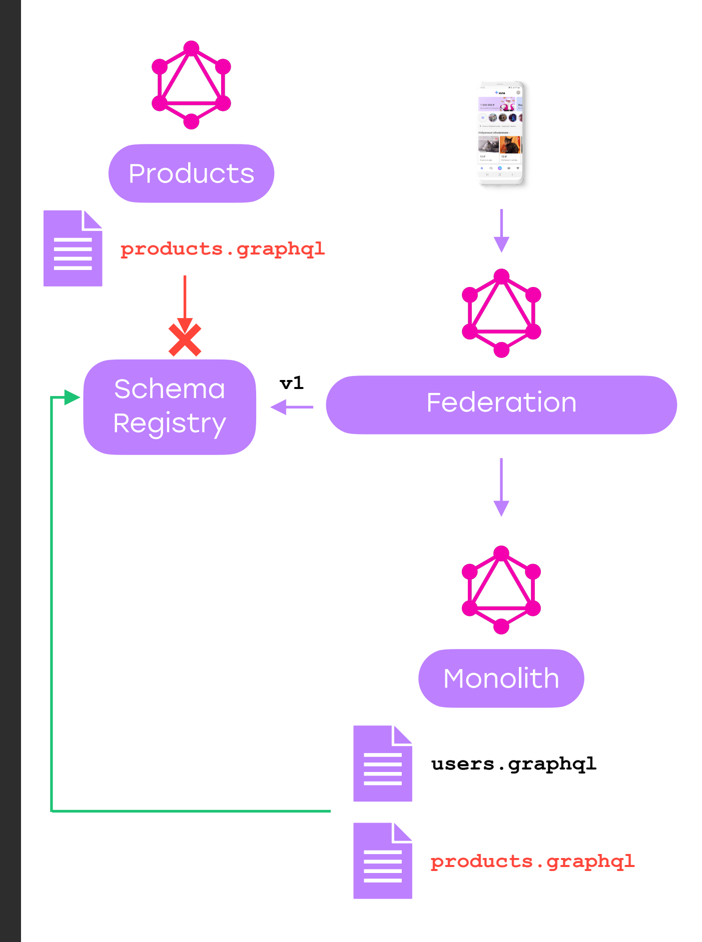
- Копируем схему с продуктами в Products сервис





Миграция схемы между сервисами

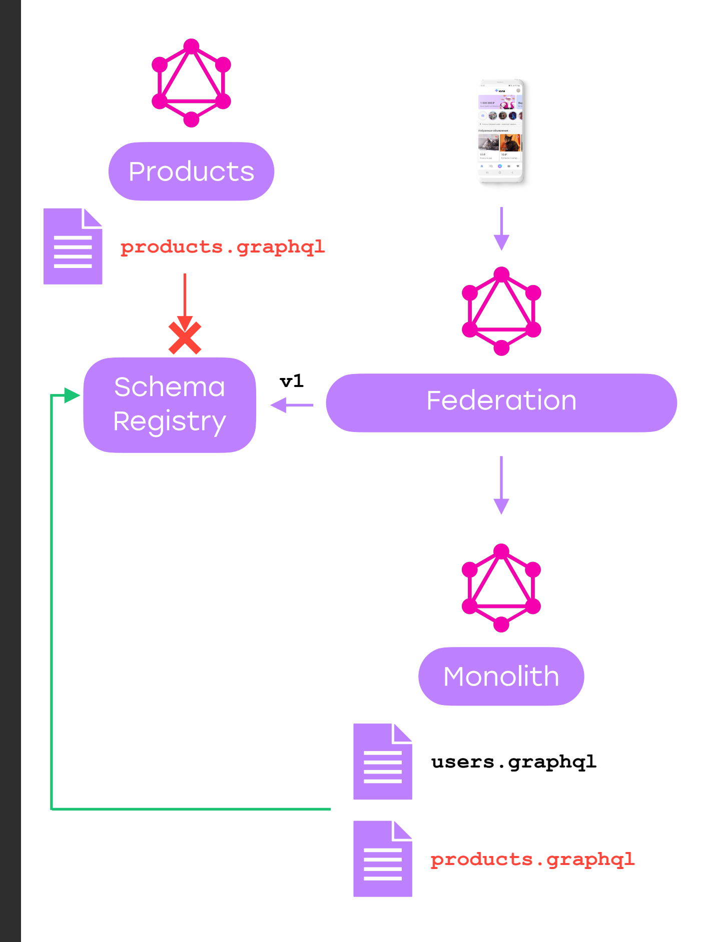
- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт





Миграция схемы между сервисами

- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт
- SR отдает предыдущую валидную схему





Миграция схемы между сервисами

- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт
- SR отдает предыдущую валидную схему
- Деплоим Products с поддержкой схемы

products.graphql

Schema Registry

v1

Federation

Monolith

Products

users.graphql

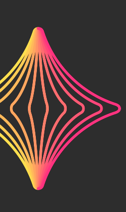
products.graphql

products.graphql



Миграция схемы между сервисами

- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт
- SR отдает предыдущую валидную схему
- Деплоим Products с поддержкой схемы
- Обновляем монолит, удаляя поддержку схемы продуктов



Миграция схемы между сервисами

- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт
- SR отдает предыдущую валидную схему
- Деплоим Products с поддержкой схемы
- Обновляем монолит, удаляя поддержку схемы продуктов
- Пушим обновленную схему монолита в Schema Registry



Миграция схемы между сервисами

- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт
- SR отдает предыдущую валидную схему
- Деплоим Products с поддержкой схемы
- Обновляем монолит, удаляя поддержку схемы продуктов
- Пушим обновленную схему монолита в Schema Registry
- Федерация забирает валидную схему, но продукты предоставляет сервис Products



Миграция схемы между сервисами

- Копируем схему с продуктами в Products сервис
- Пушим схему в SR и получаем конфликт
- SR отдает предыдущую валидную схему
- Деплоим Products с поддержкой схемы
- Обновляем монолит, удаляя поддержку схемы продуктов
- Пушим обновленную схему монолита в Schema Registry
- Федерация забирает валидную схему, но продукты предоставляет сервис Products

Schema Registry

v2

Federation

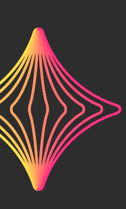
Monolith

Products

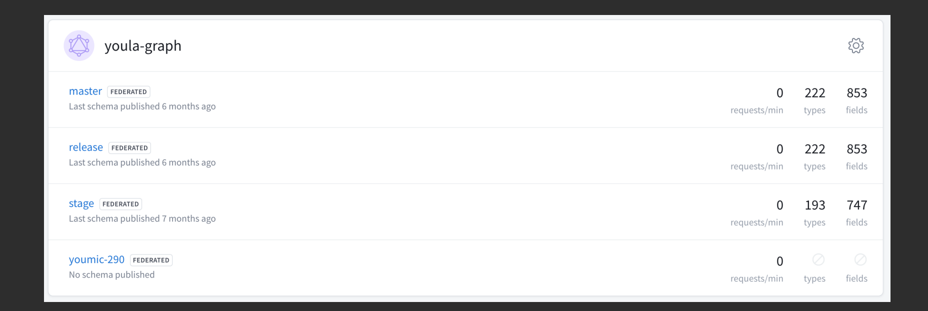
users.graphql

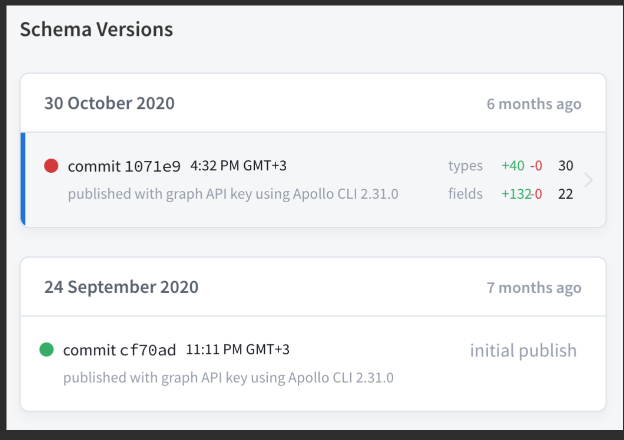
products.graphql


Apollo Studio
юла.tech



Apollo Studio





youla-graph/master

Graph Ref `youla-api-gw@master`

Version `1071e9`

227 types

853 fields



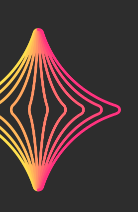
Выводы





Минусы

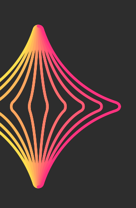




Минусы

- Сложные миграции схем





Минусы

- Сложные миграции схем
- В Федерации нет подписок (пока)





Минусы

- Сложные миграции схем
- В Федерации нет подписок (пока)
- Усложняет архитектуру сервисов





Минусы

- Сложные миграции схем
- В Федерации нет подписок (пока)
- Усложняет архитектуру сервисов
- Усложнился процесс релизов





Плюсы





Плюсы

- Заставляет более вдумчиво проектировать API





Плюсы

- Заставляет более вдумчиво проектировать API
- Дает каждой команде независимость при разработке сервиса





Плюсы

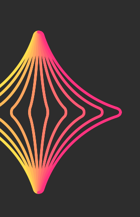
- Заставляет более вдумчиво проектировать API
- Дает каждой команде независимость при разработке сервиса
- Типизация – основа здоровых отношений клиента и сервера





Нюансики

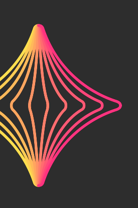




Нюансики

- Определите модератора схемы





Нюансики

- Определите модератора схемы
- Разграничьте зоны ответственности сервисов





Нюансики

- Определите модератора схемы
- Разграничьте зоны ответственности сервисов
- У Apollo нет on premise решения Schema Registry



Вопросы?



Иван Решетин

i.reshetin@corp.mail.ru

@ireshetin

Игорь Малюк

i.malyuk@corp.mail.ru

@bestmalyusha